

Your Noise, My Signal: Exploiting Switching Noise for Stealthy Data Exfiltration from Desktop Computers

Zhihui Shao*

University of California, Riverside
zshao006@ucr.edu

Mohammad A. Islam*[†]

University of Texas at Arlington
mislam@uta.edu

Shaolei Ren

University of California, Riverside
shaolei@ucr.edu

ABSTRACT

Attacks based on power analysis have been long existing and studied, with some recent works focused on data exfiltration from victim systems without using conventional communications (e.g., WiFi). Nonetheless, prior works typically rely on intrusive direct power measurement, either by implanting meters in the power outlet or tapping into the power cable, thus jeopardizing the stealthiness of attacks. In this paper, we propose NoDE (Noise for Data Exfiltration), a new system for stealthy data exfiltration from enterprise desktop computers. Specifically, NoDE achieves data exfiltration over a building’s power network by exploiting high-frequency voltage ripples (i.e., switching noises) generated by power factor correction circuits built into today’s computers. Located at a distance and even from a different room, the receiver can non-intrusively measure the voltage of a power outlet to capture the high-frequency switching noises for online information decoding without supervised training/learning. To evaluate NoDE, we run experiments on seven different computers from top vendors and using top-brand power supply units. Our results show that for a single transmitter, NoDE achieves a rate of up to 28.48 bits/second with a distance of 90 feet (27.4 meters) without the line of sight, demonstrating a practically stealthy threat. Based on the orthogonality of switching noise frequencies of different computers, we also demonstrate simultaneous data exfiltration from four computers using only one receiver. Finally, we present a few possible defenses, such as installing noise filters, and discuss their limitations.

KEYWORDS

Covert channel, data exfiltration, power analysis, switching noise

1 INTRODUCTION

The total number of registered malware samples has grown by 36% in the past year and reached an all-time high of 690 million, let alone the huge number of undiscovered malware [59]. Importantly, more than 70% of the malware threats are in the form of phishing, spywares, and Trojans that aim at stealing sensitive information, especially from end users in companies, universities, among others (which we collectively refer to as *enterprise*) [70].

In the wake of growing risks of data theft, a proactive defense is to keep sensitive data within an enterprise network at all times. Nonetheless, this approach is vulnerable to various types of covert channel attacks, through which sensitive data is stealthily transferred to a program that can access external networks and eventually send information to the outside [11, 50, 69]. For example, one

program’s usage pattern of CPU resources, if detected by another program, can be modulated for information transfer between the two [50, 57]. Consequently, to mitigate data theft risks, enterprise users commonly have restricted access to outside networks — all data transfer from and to the outside is tightly scrutinized.

Nevertheless, such systems may still suffer from data exfiltration attacks that bypass the conventional communications protocols (e.g., WiFi) by transforming the affected computer into a transmitter and establishing a covert channel. For example, the transmitting computer can modulate the intensity of the generated acoustic noise by varying its cooling fan or hard disk spinning speed to carry 1/0 bit information (e.g., a high fan noise represents “1” and “0” otherwise), while a nearby receiver with a microphone can hear the noise and decode the carried bits [8, 32, 34, 38]. Likewise, the power consumption [35, 73], the amount of generated heat [31], the electromagnetic interference (EMI) [29, 30], the system status LEDs [37, 55], and magnetic signal (to escape a Faraday cage) [36, 58] can all be modulated in a similar manner for data exfiltration.

Our contribution. We contribute to the existing body of research by designing a new data exfiltration system, called NoDE (Noise for Data Exfiltration), where a malware modulates the victim computer’s power consumption to send data over a building’s power network to the attacker’s receiver. The key novelty is that NoDE uniquely exploits high-frequency voltage ripples (i.e., electronic switching noises) generated by power factor correction (PFC) circuits built into today’s power supply units for power-consuming devices like computers. Like the existing data exfiltration attacks (Table 3), NoDE exhibits several desirable properties: a reasonable achievable bit rate (28.48 bits/s), good effective distance (27.4 meters), and no line-of-sight requirement. Additionally, based on an in-depth investigation of how PFC-induced switching noises relate to a computer’s power consumption, NoDE adds the following two distinguishing features to the literature.

- *Indirect power measurement.* NoDE uses indirect power measurement that does not require any tampering of the building’s power network and hence is more stealthy. Here, being indirect means that the target computer’s current does not directly flow through the attacker’s sensing device at the receiver; instead, the receiver only measures voltage signals containing PFC-induced switching noises which we find are correlated with the target computer’s power/current. Nonetheless, the existing power-based data exfiltration attacks rely on direct power measurement and hence are less stealthy [23, 35, 73]: a power meter is directly connected to the outlet or a sensing apparatus is placed along the cable directly powering the target device.

- *Simultaneous data exfiltration.* We identify the (approximate) orthogonality nature of PFC-induced switching noises in practice,

*Both authors contribute equally to this work.

[†]This work was partially done while M. A. Islam was at UC Riverside.

thus allowing simultaneous data exfiltration from multiple computers to a single receiver without much interference. Thus, if multiple computers within the same power network are infected, only a single receiver is needed to exfiltrate data from these computers in parallel. This results in a higher overall exfiltration rate due to the multiple parallel data streams.

More concretely, we focus on an enterprise environment, with the goal of stealthy data exfiltration from a desktop computer. Note that, we do not target military-grade systems that have sophisticated and expensive defense against information leakages (e.g., TEMPEST [26]). We first observe that the amplitude information of a computer’s electric current, and hence the power consumption, is contained in the voltage at any other power outlet connected to the same building’s power network.

In practice, however, it is very challenging to directly extract the current amplitude of the target computer from the voltage measurement, which consists of a blend of current amplitude information of all the devices within the same power network (Section 4.1). This is further compounded by the power grid’s random voltage fluctuations which can be several order-of-magnitude larger than the voltage variation caused by current amplitude variations.

We find that all desktop computers today are mandated to have built-in power PFC circuits in their power supply units to reduce harmonics [18, 43, 64]. Importantly, these PFC circuits result in prominent high-frequency current ripples between 40kHz and 150kHz [64], whose amplitude changes with the computer’s power consumption – the higher power consumption, the taller ripples, and vice-versa. These high-frequency current ripples also produce high-frequency voltage ripples at other power outlets, which are referred to switching noises. Thus, by properly filtering the received voltage signals at a power outlet, switching noises can be retained and the receiver is able to successfully extract information about the transmitter’s modulated current amplitude. Further, the switching noise frequencies are typically different for different computers and hence are not subject to much interference. This orthogonality of switching noises allows simultaneous data exfiltration attacks from multiple computers by a single receiver.

We present an end-to-end design of NoDE. Like any normal programs, NoDE only uses the transmitter’s CPU resource without any special privilege (e.g., Kernel access). Any device with data storage, such as a laptop and a cellphone, as well as an added analog-to-digital converter (ADC, to digitize the voltage signals) plugged into a power outlet can be used as a receiver. Moreover, NoDE does not require any offline training or calibration using supervised classification algorithms.

To demonstrate the practical applicability of NoDE, we run experiments on seven computers with different configurations and vendors in four different labs/offices in two separate buildings. We also achieve successful exfiltration even when four computers send data simultaneously to a single receiver. We show that NoDE achieves successful data exfiltration with an effective rate of up to 28.48 bits/second, which is reasonably high compared to many existing covert channels [28, 29, 73]. More importantly, the receiver can be located in a different room approximately 90 feet (27.4 meters) away from the transmitter. Finally, we also present a set of possible defense mechanisms, such as installing noise filters, and discuss their limitations.

2 FEATURES OF NODE AND PRELIMINARIES

In this section, we first discuss NoDE’s distinguishing features and advantages under its own subclass of power analysis-based side/covert channel attacks. Then, we provide a note on the current PFC design in power electronics.

2.1 Power Analysis-based Attacks

NoDE falls under the power analysis-based attacks, and we identify the key differences of NoDE from the existing works. With power usage information of the victim, prior studies have achieved secret key extraction from smart cards and mobile devices [6, 23, 56], anomaly detection in embedded systems [14, 54, 66], tracking websites [13, 79], among others. Besides the orthogonal context and objective, our work stands apart from the prior studies in the following key aspects.

Power measurement. A prominent assumption made by many existing attacks is that the target system’s power consumption (or current) can be directly measured [13, 79]. Nonetheless, this can only be accomplished if the sensing apparatus is placed directly inside the target system or at the nearest power outlet (indicated by Fig. 1(a) and Ⓐ in Fig. 1(b)). Moreover, the way that the transmitter’s current is measured (e.g., measuring current at Ⓑ in Fig. 1(b)) [35] results in a very low bit rate because of strong interference from other computers and/or devices.

By sharp contrast, NoDE can collect voltage signals from *any* outlet within the same power network as the transmitter, and identifies and focuses on a unique frequency band for each transmitter, thus achieving stealthy data exfiltration (even simultaneously from multiple transmitters).

Offline training. The studies on side channel attacks using power analysis are prolific, e.g., recognizing TV content out of voltage signal measurement from a nearby outlet [19] and identifying human gestures using body-induced electric signals [15]. Nonetheless, these studies typically apply supervised machine learning models and need to extract a set of features from the collected power signals and compare them against a set of pre-recorded patterns for recognition. Thus, offline model training on the target system under a controlled environment is required. Clearly, this is not feasible in our context for stealthy reasons.

Our design is fundamentally different and does not need to match received signals against pre-trained patterns, which can significantly differ from runtime conditions. Instead, to receive new information from the target computer on the fly, NoDE adapts its 1/0 bit decision threshold and filter’s passband based on a small pre-determined pilot sequence on a frame-by-frame basis.

Simultaneous transmission using orthogonal signals. In addition to the intrusive nature, direct power measurement requires dedicated power meters (i.e., receivers) for each target computer. Placing a power meter at a higher level in the power network (e.g., at Ⓑ in Fig. 1(b)) to capture multiple transmitters does not work as the current signals from the transmitters as well as other equipment are superimposed in the readings and hard to distinguish. While NoDE also collects a mix of signals from all equipment, it can still separate each transmitted signal in the high frequency (10kHz~150kHz) using the insight that PFC circuits generate orthogonal high-frequency switching noises (Fig. 12).

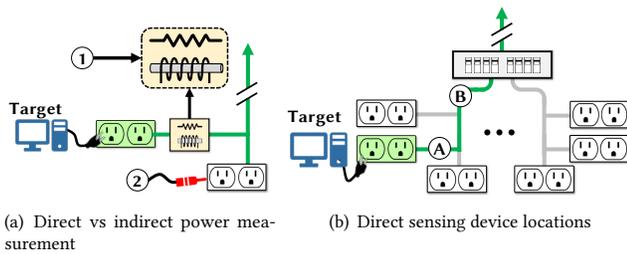


Figure 1: (a) ① is direct measurement using a sensing coil or register placed along the direct power path . ② is indirect voltage sensing. (b) Possible direct sensing device locations, such as A and B. However, location B measures superimposed power consumption of multiple outlets.

While some side channel attacks directly look at the current amplitude [13, 79], the study [20] considers a wide frequency band (e.g., 1~60kHz) of collected voltage signals. Thus, when other appliances or multiple victims are present, strong interference can be produced. For example, the attack is demonstrated on only one TV in a home or small lab with a relatively clean power network [20]. By contrast, for each transmitter, we precisely extract the frequency feature over a proper narrow band (e.g., 60Hz) without strong interference from other devices, thus achieving simultaneous data exfiltration from multiple transmitters in a large lab with about 30 active computers (Section 6.2.2).

NoDE also differs from [44] which utilizes a wider frequency band (i.e., 1kHz or more) and estimates a data center-wide *aggregated* power consumption over a much lower time resolution (i.e., once every minute) for load injection attacks. Whereas, NoDE is specifically designed to detect individual power consumption at a time resolution of milliseconds.

2.2 NoDE and TEMPEST

Electromagnetic emission has long been known as a major source of information leakage [41]. Notably, the eavesdropping on electromagnetic radiation (EMR) of communication equipment in the early-mid 1900s leads to the development of, to date partially classified, defense technologies code-named TEMPEST [26]. The recently declassified TEMPEST defense imposes stringent restrictions on electromagnetic radiation from computer systems both over the air and through power lines [26]. However, TEMPEST defense, shielding equipment and/or Faraday caging, is very expensive and mainly used in military application. For example, NATO countries spend billions of dollar of their defense budget each year for TEMPEST shielding [1]. While the defense requirement is outlined, the technology for TEMPEST attacks still remains classified. There is skepticism on the feasibility of TEMPEST attacks [26, 40], not to mention the sophisticated sensing equipment necessary to carry out such an attack. Consequently, typical enterprise environments (e.g., companies), which are our target systems, may not necessarily adhere to the costly TEMPEST defense practices.

A relatively inexpensive alternative to TEMPEST using software, named Soft-TEMPEST is proposed in [48]. Soft-TEMPEST targets obsolete CRT monitors and mainly reduces the electromagnetic

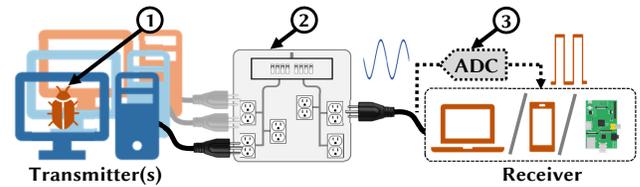


Figure 2: Threat model. ① Modulation program in the transmitter. ② Building's power network. ③ An analog-to-digital converter inside an innocuous-looking device.

emission distance, whereas NoDE utilizes power networks for data exfiltration at a distance.

2.3 Power Electronics

It is well known in the field of power electronics that the high-frequency switching operation in PFC circuits produces voltage noises and is fundamental for improving the power factor of appliances [64, 80]. The existing research on PFC designs is primarily from the *efficiency* perspective, e.g., how to select the conduction mode and switching frequency to achieve the best energy efficiency of power supply units and meet regulation compliances [64]. In sharp contrast, there is much less understanding from the *adversarial* perspective: *how can the PFC-induced switching noises be exploited as useful signals for stealthy data exfiltration attacks?* NoDE fills the gap by performing an in-depth study of how the switching noise amplitude relates to a computer's power consumption and uniquely transforming switching noises of a computer's power supply unit into data-carrying signals for data exfiltration.

3 THREAT MODEL

We consider a broadly-interpreted enterprise environment (such as company and university) which is a primary target for data theft [12, 77]. We focus on a desktop computer, because it is the predominant type of computer (especially for storing important data) and its built-in PFC circuit is suitable for information transfer (Section 4.3). For brevity, we also use “transmitter” or “computer” to denote the transmitting desktop computer.

Our threat model is illustrated in Fig. 2, including one or more transmitters and a receiver. Both the transmitters and receiver are connected to the same building's power network. Note that, because typically there are filters between different buildings' power networks, even dedicated PLC adapters cannot reliably communicate across different buildings [7].

Transmitter. A transmitter is a desktop computer infected by malware that intends to send sensitive information (e.g., password and financial information) to the outside without using any network or removable storage. Like in the existing literature on covert channels, our threat model builds upon the malware's capability of obtaining sensitive information, and is not intended for sending large files due to rate limits. The malware can use the transmitter's CPU like any normal programs, but no special privileges are assumed by NoDE. Importantly, while it leverages the building's power network, NoDE does not need to intrusively install a new or use an existing dedicated PLC adapter in the target transmitter.

Receiver. The receiver can be any innocuous-looking device that is plugged into an outlet in the same building’s power network as the transmitter. The receiver needs an ADC for digitizing its received voltage, which is universally used by any signal-collecting digital systems (e.g., digital temperature sensor) and can be easily hidden inside a laptop/cellphone charger. Moreover, the receiver can be located in a distant room different than the transmitter. Thus, there is no prohibitive requirement for a receiver. Even though the building is solely occupied by a single enterprise, guests are typically still allowed to plug their laptops or cellphones into an outlet for charging.

Practicality of malware injection. While there are various ways that malware can get into a computer, we classify them into two categories – easy and hard – based on the level of difficulty and effort required for malware infection.

- *Easy:* We consider the malware injection “easy” when the target computer can take input from the outside through external media and/or networks. For example, the target computer can get infected with malware when it visits malicious webpages or is connected to an affected USB drive, during which malware can be implanted without being noticed. It is the predominant approach of malware injection for today’s enterprise environments [70]. Not to mention those malware injection incidents affecting millions of Internet users every day [59, 70], a striking example of infecting a mission-critical system is that the malware Stuxnet infected Iran’s nuclear program through an infected USB drive [49, 51].

- *Hard:* We consider that the malware injection is “hard” when there is no easy approach to malware injection. For example, the target computer can be almost completely isolated from all external networks, which is also known as “air-gaping” [28]. In such a scenario, malware can still be injected by exploiting hardware/software backdoors throughout the supply chain. For example, it has been recently reported that some microchips were added to servers’ motherboards during the manufacturing process without the knowledge of a major server vendor [63]. In other instances, ShadowPad was implanted into a software program developed by a third-party vendor, affecting hundreds of large businesses [45], while malicious batteries [53] and other hardware Trojans [76] are all known threats to data security. An even more striking example is that over 100,000 computers that had *never* been connected to any network were also implanted with information-stealing hardware Trojans by using a classified technology [68].

A complete data exfiltration attack involves two major steps. The first step is to inject a malware program into the target computer and collect sensitive information. The second step is to establish a covert communication channel to send sensitive data to the receiver. Like the existing covert channel literature [28, 33, 35, 53], our work focuses on the step of establishing a covert communication channel. Thus, we can embed NoDE into an existing information-collecting malware. Concretely, we describe the procedure of integrating NoDE with malware created by the recently released malware toolkit L0rdix [2, 46] as follows. The L0rdix toolkit comes with a variety of pre-built and configurable functionalities. It can steal information from the victim system by collecting login information, browser cookies, and files matching pre-configured extensions. It can also monitor clipboard content to steal data matching predefined strings. L0rdix toolkit offers botnet capabilities such as

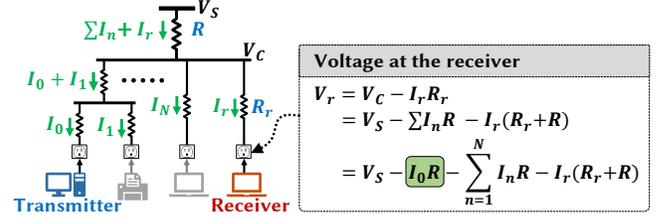


Figure 3: Illustration of a building’s power network topology showing possible locations of transmitter and receiver. The transmitter current I_0 is embedded in receiver’s voltage V_r .

opening a specific URL, execute commands, kill processes, upload and download files and run executable files. L0rdix also comes with USB infection, mining, detection prevention, anti-analysis, and anti-VM capabilities as well. In our context, we leverage the malware’s existing capabilities of system infiltration and information collection, and utilize its botnet functionality with our covert channel communication. We can either attach our code with L0rdix malware or configure L0rdix to download our program as malicious code. Other malware generators such as Senna Spy FTP, which spread as a Trojan bundled with free software, can also be used to secretly download the code of NoDE into an infected system [71, 75].

4 POWER NETWORK AS A COVERT CHANNEL

In this section, we present our covert channel residing in a building’s power network.

4.1 Overview of a Building’s Power Network

The utility power typically enters a building through a single point. Then, through a distribution box, power is split to different floors/rooms in parallel and finally to different power strips/outlets. For large buildings, a multi-level distribution hierarchy may be used. Further, the utility may provide three-phase power to buildings, and the three phases can be divided depending on functions (e.g., offices on one phase, and the central air conditioner on another) or physical topologies. For illustration, Fig. 3 provides a simplified example of a single-phase power network of a building, highlighting the parallel connection of different power outlets. The common source voltage V_S enters a distribution/panel box and reduces to V_C due to a voltage drop caused by the line resistance. Then, the voltage V_C is supplied to different rooms/outlets. We can write the receiver’s voltage signal V_r as

$$V_r = V_C - I_r R_r = V_S - I_0 R - \sum_{n=1}^N I_n R - I_r (R_r + R), \quad (1)$$

where R is the resistance of the common line from which all currents flow, and R_r is the resistance of the line directly supplying power to the receiver. Clearly, the receiver’s voltage signal V_r contains the transmitter’s current denoted by I_0 in Fig. 3, whose amplitude can be modulated by varying the CPU load to carry information (Section 5.1). Thus, if the receiver is able to extract $I_0 R$ out of its signal V_r , it can exfiltrate information from a computer through the power network.

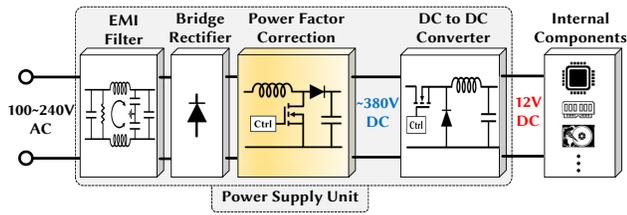


Figure 4: Components of a desktop computer’s power supply unit.

4.2 Computer’s Power Supply Unit

We now look at the anatomy of a computer’s power supply unit and identify an important component – PFC circuit – which generates high-frequency current ripples that can be detected by the receiver.

4.2.1 A closer look at computer’s power supply unit. As shown in Fig. 4, a power supply unit based on the standard SMPS design draws 100~240V AC voltage from an outlet and then, after multiple stages, provides regulated 12V DC voltage to internal components such as CPU. Specifically, at the front end, there is an EMI filter to limit frequency components of greater than 150kHz both coming from and conducted back to the power network, in compliance with international regulations [17, 39]. Then, a rectifier converts the incoming AC voltage to a pulsating DC voltage (unipolar half-sine waves), followed by a PFC circuit which improves the power factor by regulating the input current waveform and making it resemble the entering voltage’s sine wave. Fig. 5(a) shows a snapshot of the current waveform drawn by our Dell computer with PFC. The PFC elevates the voltage to around 380V, which is stepped down and becomes 12V DC voltage for internal components.

Harmonic distortion is undesirable since it reduces the power factor and causes unwanted power losses in the power system [64]. Low-power devices with SMPS (switch mode power supply) are allowed to have a low power factor without PFC (see Fig. 22 in Appendix C.3 for the current waveform). Nonetheless, regardless of the actual power consumption, all devices with a power rating of 75W (applicable for desktop computers) must have PFC circuits for mitigating harmonics as mandated by international regulations [18, 43, 64]. Thus, a crucial point we highlight is that *the PFC requirement for mitigating harmonics universally applies to all of today’s desktop computers*.

4.2.2 Frequency spikes generated by PFC. As shown in Fig. 5(a), while improving power factor, the PFC circuit also produces high-frequency current ripples due to its working principle [64]. Specifically, the core of a PFC circuit is rapidly switching the incoming current between two modes – a rising mode where the current increases, and a falling mode where the current decreases. Through switching, the current drawn from the power outlet resembles a sine waveform following the voltage signal. The switching frequency is determined by a controller as well as the PFC components (e.g., inductor), and typically falls into the range of <40kHz, 150kHz>, which is not subject to EMI regulations that set limits on frequencies greater than 150kHz [17, 64].

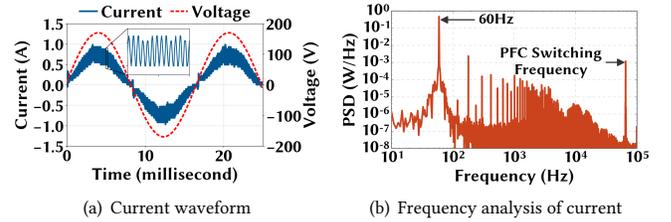


Figure 5: (a) Current drawn by our Dell PowerEdge computer. (b) The current has a high-frequency spike due to PFC operation under CCM.

There are three basic modes for PFC switching (a.k.a., conduction mode) as summarized in Table 4 in Appendix C.1. In practice, vendors may adopt proprietary designs using variants of the basic modes [64]. Naturally, the PFC’s switching operation results in high-frequency current ripples, thus generating a PSD (power spectrum density) spike around the switching frequency. Illustrative current waveforms and frequency analysis results are shown in Table 4. Note that when PFC is not used, there exist no such PSD spikes within <40kHz, 150kHz> (see Fig. 22(b)).

Desktop computers’ power supply units almost all have a rated capacity of over 300W to accommodate extensibility. Thus, the continuous conduction mode (CCM) is most widely-used due to its low peak current. We show the current and its frequency analysis for our Dell computer in Fig. 5. It can be observed that with PFC, the harmonics are an order of magnitude smaller than the 60Hz component. Furthermore, the current waveform and frequency analysis match with those illustrative figures for CCM in Table 4 in Appendix C.1, clearly showing a prominent high-frequency PSD spike generated by the PFC circuit.

4.3 Extracting Transmitter’s Current based on PFC Switching Frequency

The receiver’s voltage signal in Eqn. (1) contains all the frequency components of the common source voltage and currents of all connected devices. That is, the high-frequency current ripples generated by PFC circuits affect voltage signals at any power outlet within the same power network, which are referred to as switching noise [64]. Consequently, if we filter out all but the transmitter’s high-frequency switching noises from the receiver’s voltage V_r , we are left with the transmitter’s switching noise around its PFC frequency and the switching noise amplitude is also highly correlated with the transmitter’s current I_0 . This is achieved based on the (approximate) orthogonality of PFC’s switching noises.

Different computers typically have *non-overlapping* PFC switching frequencies. In fact, our experiment shows that even for computers with the same configuration and manufactured by the same vendor, different computers still have (slightly) different PFC switching frequencies due to manufacturing process variations and can simultaneously transmit data without much interference. Furthermore, the prominent PSD spikes between 40kHz and 150kHz do not interfere significantly with harmonics (predominantly less than 20kHz) generated by other devices. Therefore, in practice, the ripples in the

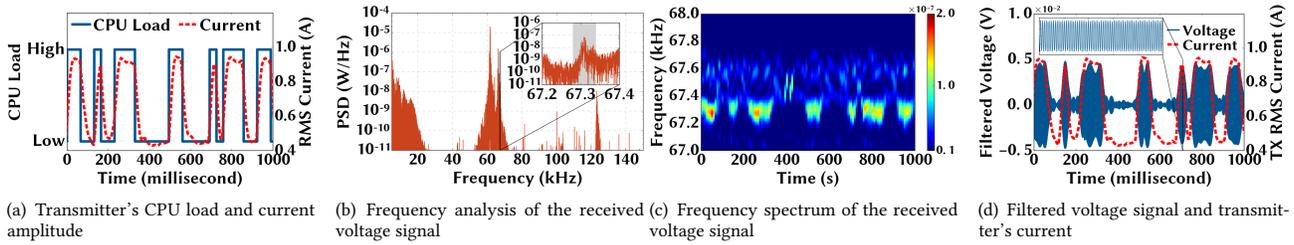


Figure 6: Experiment in Lab #1 with transmitter and receiver separated by 55ft. By applying a filter with passband of $\langle 67.28\text{kHz}, 67.34\text{kHz} \rangle$, the amplitude of filtered voltage signals acquired by the receiver matches the transmitter's current amplitude.

receiver's voltage signal caused by the target transmitter's switching noise do not suffer from significant interferences from other sources.

We empirically validate the feasibility of extracting the transmitter's current amplitude information. Our experiment is conducted in a lab with 30+ computers, where the transmitter and receiver are plugged into two outlets located about 55 feet away from each other. The details of the setup are presented in Section 6.1. We vary the transmitting computer's current by varying its CPU load because compared to components such as hard disk and memory chips, a computer's CPU has a high dynamic power that can be easily adjusted by loading/unloading the CPU. Moreover, CPU is ubiquitously available in all computers and needed by any running program. As GPUs are power-consuming, a computer's current can also be significantly varied in a similar fashion by using increasing the utilization of a dedicated GPU. Note, however, that a GPU is less ubiquitous compared to a CPU, especially in ordinary office environments. Thus, throughout our study, we will only utilize CPU to change a computer's current for data exfiltration.

We show in Fig. 6(a) the transmitter's CPU load and current amplitude, which match with each other quite well. Then, we perform a frequency analysis of the received voltage signal and show the result in Fig. 6(b). We see large frequency components between 40kHz and 80kHz (and sporadic higher-frequency spikes). These are mainly due to different computers' PFC switching operations, and the components around 67.3kHz are caused by our transmitter. The temporal variation of the PSD spikes created by the transmitter is shown in the frequency spectrum in Fig. 6(c) where we can easily identify the transmitter's high current periods. Next, we filter the collected voltage signal with a passband of $\langle 67.28\text{kHz}, 67.34\text{kHz} \rangle$ and show the filtered voltage signal in Fig. 6(d) where the filtered voltage signal resembles the current ripples (as shown in the zoom-in window). The filtered voltage amplitude is close to zero during the low current periods because the PSD spikes shift away from the 60Hz passband.

In summary, we have demonstrated that, with a proper band-pass filter, the amplitude of the receiver's filtered voltage signal can recover the transmitter's modulated current amplitude and hence be exploited for demodulation.

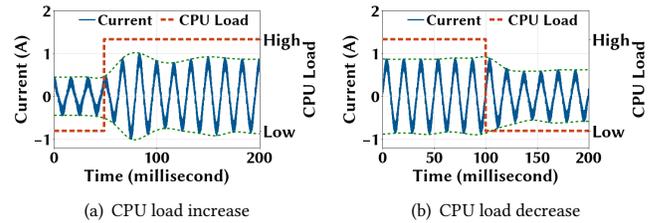


Figure 7: Input current's response to CPU load changes. Due to power supply unit's internal control, the current does not change immediately after change in the CPU load.

5 THE DESIGN OF NODE

In this section, we present the design of NoDE. As shown in Fig. 8, NoDE includes both a transmitter (i.e., a desktop computer with implanted malware) and a receiver (i.e., any voltage-collecting device plugged into a power outlet).

5.1 Transmitter Design

Like in the prior literature on covert channels [28, 29, 73], NoDE focuses on the *physical* process of data exfiltration — converting 1/0 bits into current amplitudes and decoding it from a remote outlet. That is, the implanted malware already collects needed information (Section 3) and encodes it into 1/0 bit streams that are ready for data framing and modulation. Next, we address the key design issues.

5.1.1 Choosing symbol rate. NoDE modulates the transmitter's current amplitude by varying the CPU load based on 1/0 bit values. Thus, the achievable symbol rate crucially depends on how fast the current amplitude changes in response to the CPU load. While the CPU usage can be adjusted within a millisecond or even faster [60], the current drawn by a computer may not instantly follow (i.e., lags) the CPU usage. In our experiment with a Dell PowerEdge computer in Fig. 7(a) shows that the current takes about 20~30 milliseconds to reach the peak after jump in the CPU load at around 50 milliseconds. In addition, because of the feedback control adopted by a power supply unit, a sudden CPU load change can create a current inrush followed by a current dip. As shown in Fig. 7(b), the current also lags by around 30 milliseconds when the CPU load sharply changes from high to low.

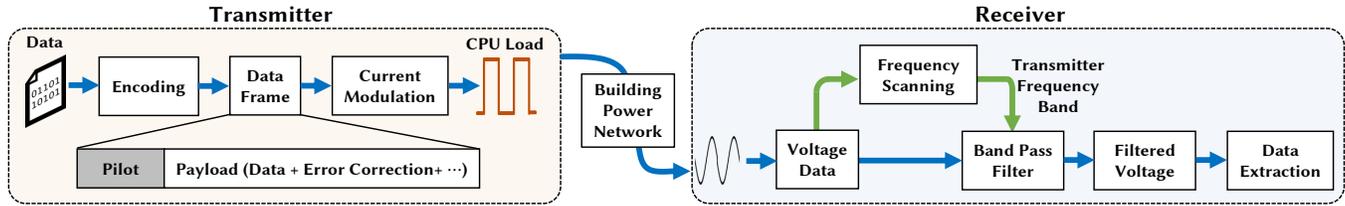


Figure 8: Block diagram of transmitter and receiver in NoDE. The transmitter sends data to the receiver over the power network.

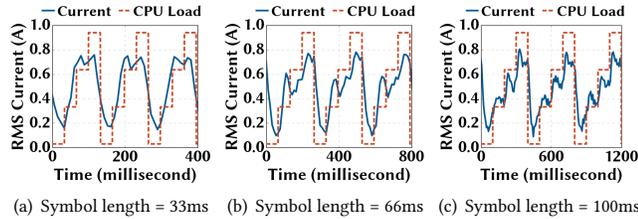


Figure 9: Experiments for 2 bits per symbol on our Dell PowerEdge computer. With a symbol length of 100ms, there is sufficient time for the input current to steadily respond to CPU load changes.

Based on our experiment results, we choose 33ms as the default symbol duration. This means that, *by modulating the transmitter’s current amplitude, the achievable symbol rate in NoDE is 30 symbols per second.*

5.1.2 Choosing modulation mode. For a fixed symbol rate, a higher bit rate can be achieved if each symbol carries more bits. Here, we run experiments to see how many bits can be successfully mapped into each symbol: 2^N discrete current/power levels are needed for N bits per symbol. For $N = 2$, we vary CPU loads at four levels (0%, 25%, 75% and 100%, representing “00”, “01”, “10” and “11”, respectively) and see how input current changes, given three different symbol lengths — 33ms, 66ms, and 100ms.

We show the results in Fig. 9 and see that when each symbol lasts 33ms, there is a mismatch between the CPU load (representing 2 digital bits) and the resulting current amplitude, thus clearly leading to a very high symbol error rate. As explained in Section 5.1.1, this mismatch is mainly due to the complex feedback control mechanisms and large capacitors inside a computer’s power supply unit. If the symbol length increases to 66ms, there is still insufficient time for the current amplitude to yield a steady response. The current amplitude correlates well with the CPU load when the symbol length increases to 100ms, but this means that the effective bit rate is only 20 bits/second, which is even lower than 30 bits/second achieved by a binary modulation with a symbol length of 33ms. With $N > 2$ bits per symbol, there is an even poorer correlation between the CPU load and input current, unless the symbol length is sufficiently large.

As a result, we choose *binary* modulation and use high and low currents to represent “1” and “0”, respectively. Concretely, we build upon the existing literature [34, 73] and design a simple current modulator as described in Algorithm 3 in Appendix D. The current

modulator program takes 1/0 bit information as the input and runs some dummy calculations (e.g., generating random numbers) to load the CPU and change the computer’s input current.

5.1.3 Choosing the frame length. Like in many communications systems [25], NoDE groups 1/0 bit sequences into frames, each beginning with a pilot sequence (Section 5.1.4). As shown in Fig. 8, following the pilot symbols is the actual payload that contains uncoded bits or coded bits using error correction techniques.

In our context, the transmitter’s PFC switching frequency is unknown to the receiver and may vary over time, albeit slowly. Throughout each frame, however, the transmitter’s PFC switching frequency should remain relatively constant. As shown in the cumulative density function (CDF) and the frequency spectrum in Fig. 21, the PFC switching frequency does not vary by more than 50Hz within 5 seconds. On the other hand, for a good bit detection, the receiver’s frequency band for filtering voltage signals can be only 60Hz. Thus, we conservatively choose a frame length of 100 symbols, resulting in frame duration of 3.3 seconds.

5.1.4 Choosing pilot symbols. In wireless communications, pilot symbols are symbols mutually known to both the transmitter and receiver and inserted at the beginning of each frame, allowing the receiver to estimate the channel state and synchronize data reception [25]. In our context, pilot symbols are needed by the receiver to identify the transmitter’s signals and the start of a data frame (details in Section 5.2). Nonetheless, the length of a pilot sequence needs to be properly chosen. A too short sequence may not be enough for the attacker to accurately acquire the transmitter’s PFC switching frequency, whereas a too long sequence takes up an unnecessarily high overhead. In our experiments in Section 6.2.3, we find that a 6-bit pilot sequence (“110010” in our study) strikes a good balance, while we also examine 4-bit and 8-bit pilot sequences (Table 1). Note that, for some scenarios (e.g., when there are multiple transmitting computers), a longer pilot sequence may be needed to uniquely identify each transmitting computer and better locate the corresponding frequency band. However, in general, there is no need to change the pilot sequence for different buildings or power suppliers.

To summarize, considering the hardware constraints and behavior of the PFC switching frequency, we choose in our default design a symbol period of 33 milliseconds, 1 bit per symbol, 100 bits in each data frame with a frame duration of 3.3 seconds, and a 6-bit pilot sequence “110010”.

Algorithm 1 Finding a Passband for Filtering Voltage Signals

```

1: Input: Voltage signal  $V(t)$  of a pilot length, symbol length  $T_b$ ,
   pilot sequence  $\mathbf{B}_p$ , max window  $F_{\max}$  kHz, increment  $F_{inc}$  kHz
2: for  $lb = 20 \sim (150 - F_{\max})$  kHz with increment  $F_{inc}$  do
3:   for  $ub = lb + F_{inc} \sim lb + F_{\max}$  with increment  $F_{inc}$  do
4:      $\tilde{V}(t)_{lb,ub} \leftarrow V(t)_{\langle lb,ub \rangle}$  //filtering
5:      $E(t)_{lb,ub} \leftarrow \text{Envelop of } \tilde{V}(t)_{lb,ub}$ 
6:     Extract bits  $\mathbf{B}^*$  from  $E(t)_{lb,ub}$  based on  $T_b$ 
7:     Error vector  $\mathbf{err}(t)_{lb,ub} \leftarrow \mathbf{B}^* \text{ XOR } \mathbf{B}_p$ 
8:   end for
9: end for
10: if  $\mathbf{err}(t)_{lb,ub} == \mathbf{0}$  then //pilot sequence found
11:   Return  $\langle lb, ub \rangle$ 
12: end if

```

5.2 Receiver Design

On the receiver side, NoDE calculates the average amplitude of filtered voltage signals for every bit length and compares it against a detection threshold for deciding a 1/0 bit value. There are three major steps: (1) finding a passband for the filter; (2) identifying the start of a data frame; and (3) demodulating the extracted signals into 1/0 data bits.

5.2.1 Finding the filter’s passband. In practice, the transmitter’s PFC switching frequency is unknown to the receiver. Based on a predetermined pilot sequence, we propose a scanning process to find a frequency passband for filtering received voltage signals and retaining the prominent PSD spikes generated by the transmitter. The scanning process is described in Algorithm 1, where two nested loops scan through the frequency bands from 20 kHz to 150 kHz with a moving window of variable size. The notion of $\langle lb, ub \rangle$ means a band-pass filter with lb and ub being the lower and upper cutoff frequencies, respectively. When $\langle lb, ub \rangle$ appears as a subscript, it means that a signal passes through a band-pass filter $\langle lb, ub \rangle$.

In each inner loop, the amplitude/envelop of filtered voltage signal is extracted and then evenly sliced into bit-length pieces. Then, the average amplitude for each piece is used for deciding binary bit values. Specifically, we set the the mean of bit-wise average amplitudes of the filtered signal as the binary bit decision threshold: a bit-wise average amplitude higher than the threshold is decoded as “1”, and “0” otherwise. Then, by comparing the extracted bits with the pilot sequence, we return the passband that yields no errors for pilot detection.

As the PFC switching frequency does not vary significantly over time, we only need to scan through a wide frequency range (e.g., from 20kHz to 150kHz) once. *After the initial scan, the receiver only needs to quickly fine tune its filter’s passband over a much narrower range* (e.g., 500Hz around the previously-found passband, instead of 20kHz to 150kHz) to compensate for runtime switching frequency offsets. Moreover, provided that voltage signals are acquired and stored, bit extraction can be done offline, and hence the scanning complexity is not an issue. In fact, with $F_{inc} = 0.01\text{kHz}$, Algorithm 1 only takes less than 5 minutes in Matlab on a laptop.

Algorithm 2 Identifying Frame Start and Bit Threshold

```

1: Input: Pilot length  $T_p$ , bit length  $T_b$ , pilot sequence  $\mathbf{B}_p$ 
2: loop at current time  $t$ 
3:   Obtain filtered voltage  $\tilde{V}(t, t - T_p)$ 
4:    $E(t, t - T_p) \leftarrow$  Signal envelop of  $\tilde{V}(t, t - T_p)$ 
5:   Extract bits  $\mathbf{B}^*$  from  $E(t, t - T_p)$  based on  $T_b$ 
6:   Error vector  $\mathbf{err}(t) \leftarrow \mathbf{B}^* \text{ XOR } \mathbf{B}_p$  //bit-wise
7:   if  $\mathbf{err}(t) == \mathbf{0}$  then //pilot sequence found
8:     Return payload starting time  $t$ , and bit detection thresh-
       old (average of  $E(t, t - T_p)$ )
9:   end if
10: end loop

```

5.2.2 Identifying the start of a frame. While the filter’s passband does not vary significantly due to the slow variation of PFC switching frequencies, the bit detection threshold depends on the non-controllable environment (e.g., other computers with similar PFC switching frequencies can cause interferences) and hence can change quickly at runtime. We propose a time-domain scanning process to identify the bit detection threshold and start of each data frame based on the predetermined pilot sequence. The scanning process is presented in Algorithm 2.

5.2.3 Extracting data bits. After identifying a pilot sequence, the receiver can extract actual payload bits using the bit detection threshold returned by Algorithm 2: if a bit-wise average amplitude of the filtered signal is higher than the threshold, then the corresponding payload bit is decided as “1”, and “0” otherwise.

6 EVALUATION

This section presents experiment results to validate the practical feasibility of NoDE, highlighting that NoDE achieves an effective rate of up to 28.48 bits/second to a receiver located in another room about 90 feet (≈ 27.4 meters) away.

6.1 Methodology

Experiment setup. As listed in Table 2, we test seven computers with different operating systems/configurations, including four recently purchased from two top vendors (Dell and Acer) which collectively account for more than 25% of the global PC market share [42], an iMac (27 inch), and two custom-built computers with top-brand power supply units (Corsair and EVGA). They all have CCM type PFC in their power supply units. Our collection of desktops is representative of commonly used models in our target case of modern enterprise office setup.

These computers are located in four different labs/offices in two different buildings (referred to as A and B, respectively). As our default location, Lab #1 in Building A is a large shared lab space housing about 30 students with 30+ *active* desktop computers, where the transmitter’s and receiver’s power outlets are approximately 55 feet away from each other. Thus, Lab #1 represents an environment with high power line noises. The layout of Lab #1 is illustrated in Fig. 10. In addition, we also run experiments in another two labs (#2 with about 10 students and #3 with about 15 students) in Building A. Our experiment setup in Building B is illustrated in Fig. 14, where

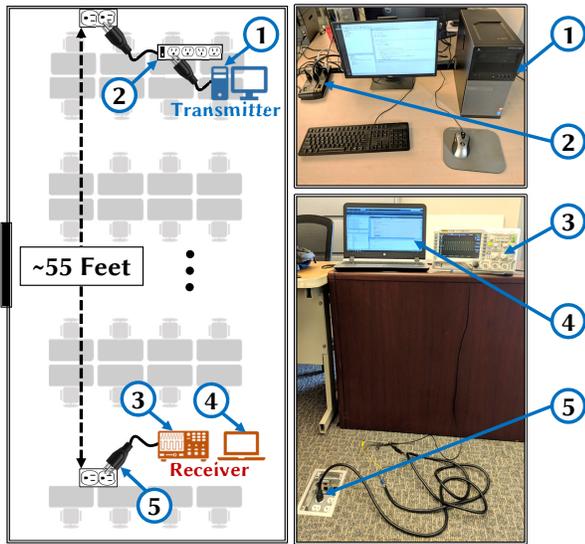


Figure 10: Experiment setup at Lab #1 in Building A. ① Transmitter. ② Power strip. ③ ADC for voltage signal acquisition (oscilloscope in our experiments). ④ Laptop for voltage filtering and bit detection. ⑤ Receiver’s power outlet.

the transmitter and receiver are located in two different rooms about 90 feet away from each other.

Transmitted data. Like other studies on covert channels [28, 29, 73], NoDE focuses on stealthy exfiltration of already collected information. Thus, for illustration, we use 8-bit ASCII values of the string “password123” followed by six random bits as the payload data and “110010” as the 6-bit pilot sequence. Each frame has 100 bits. We also randomly generate 50 data frames for each experiment. Each symbol carries one bit and lasts 33ms. Like the existing covert channel literature, error correction coding is orthogonal and omitted in our experiments.

Current amplitude modulation. To modulate the current amplitude (Section 5.1.2), we implement a current modulator following Algorithm 3 which can be incorporated into existing malware for data exfiltration. The program is written in Java and uses multiple nested $\sin()$ and $\cos()$ computations on a random floating number to vary the CPU load. We use multi-threading to increase the impact on the CPU. However, as the default case, we do not use any CPU pinning and allow regular OS scheduling.

Voltage measurement. To collect voltage signals, we use a Rigol 1074z oscilloscope as a proxy ADC circuit. To remove the voltage’s dominant 60Hz component and improve the signal acquisition precision, we follow [20] and insert a RC high-pass filter with ~ 10 kHz cutoff frequency between the power outlet and the oscilloscope. We use 500 kilo-samples per seconds (kSa/s) as the default sampling rate. While we use a bulky oscilloscope for convenience, the receiver only needs to hide a small ADC circuit (e.g., using ATmega Microcontroller) to acquire and digitize the voltage signal (Section 3).

Frequency analysis and filters. We use Matlab to perform Fourier analysis and signal filtering. In real-world implementation,

the receiver may also use a digital signal processing chip to filter voltage signals and demodulate bit values in real time.

Metrics. We calculate the bit error rate (i.e., percentage of bit errors) and effective transmission rate measured in bits/second (i.e., actual payload bit rate, excluding pilot symbols and erroneous bits). We also list in Table 2 the distance between the transmitter’s and receivers’ power outlets.

6.2 Evaluation Results

We now present our evaluation results, highlighting that NoDE achieves *stealthy* data exfiltration from desktop computers. We first present a snapshot of data exfiltration using NoDE followed by the case where four transmitters are simultaneously sending data to a single receiver. We then evaluate NoDE under different background applications, different numbers of CPU cores used by NoDE, and different pilot lengths. We also test how the CPU states affect NoDE, and the effectiveness of NoDE in data exfiltration without a line of sight. Finally, we present results with computers from different manufacturers.

6.2.1 A snapshot of data exfiltration. We show in Fig. 27 the different stages of NoDE for data exfiltration from our Dell Optiplex computer with 4 CPU cores. For clarity, we only show the first 50 bits in a data frame. At the top, we show the frame bits with the pilot sequence highlighted in a gray shade. The “X”s prior to the pilot sequence indicate “no data”. Also, the time index “0” indicates the start of the frame. We then show the transmitter’s current, which is modulated by varying the CPU load. We can see that the current amplitude changes with the transmitted bits.

Then, we show the receiver’s unfiltered voltage signal, which is affected by grid voltage variations as well as all other loads sharing the same power network and hence barely reveals any useful information. Next, we show the filtered voltage signal with a passband of $\langle 67.28 \text{ kHz}, 67.34 \text{ kHz} \rangle$ identified by Algorithm 1. The envelop/amplitude of the filtered voltage signal is extracted, and then the average amplitude is used for bit detection. From the pilot sequence, we identify the frame’s starting point and set the bit detection threshold accordingly (Algorithm 2). The bit-wise average amplitudes are then demodulated into the received bits — an average amplitude above the detection threshold is considered as “1”, and “0” otherwise. In this experiment, the received bits perfectly match the transmitted bits without any errors, resulting an effective payload bit rate of 28.48 bits/second.

6.2.2 Simultaneous transmissions. We simultaneously use multiple computers as transmitters to demonstrate that the PSD spikes from different computers do not interfere with each other. We conduct this experiment in Lab#1 using three Dell OptiPlex desktops with identical power supply units and one Dell XPS desktop with a different supply unit. Note that, while we use four computers as transmitters, there are other computers (30+) in the building that also generate switching noise spikes due to their PFC circuits.

Fig. 12 shows the frequency spectrum of the voltage signal from the receiver while the corresponding data frame extractions are shown in Appendix E.2. In the top figure, we show the frequency spectrum of the three transmitters with identical power supply units whose PSD spikes are close to each other (within the frequency

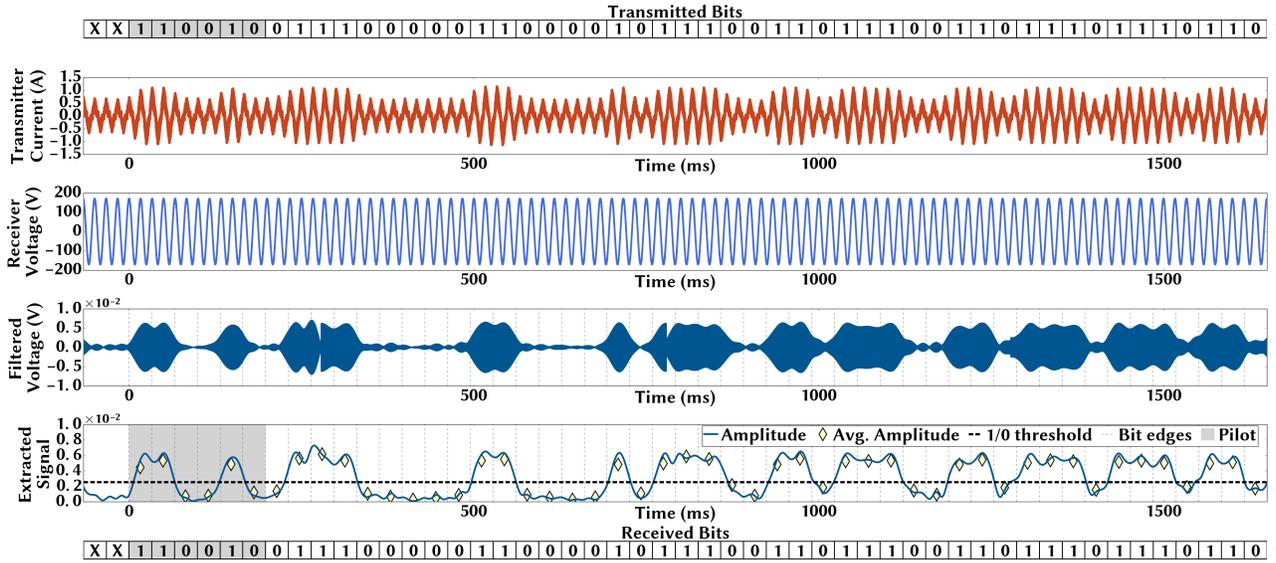


Figure 11: Dell Optiplex computer: A snapshot of data exfiltration. The receiver’s voltage signal is filtered with a passband of $\langle 67.28\text{kHz}, 67.34\text{kHz} \rangle$ and the bit-wise average amplitudes of filtered voltage signals reveal the transmitted bits.

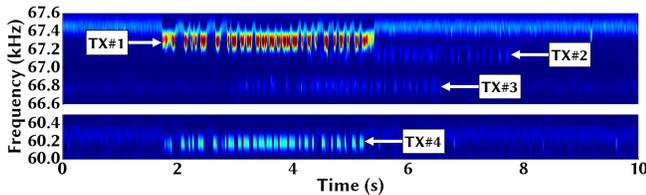


Figure 12: Four transmitters (TX#1~4) simultaneously sending data to a single receiver. Bit error rates of TX#1, TX#2, TX#3, and TX#4 are 0.0%, 6.8%, 1.1%, and 0.0%, respectively. The higher bit error rate for TX#2 is due to the partial overlap with TX#1.

band $\langle 66.6\text{kHz}, 67.6\text{kHz} \rangle$). The bottom figure shows the frequency spectrum of the other transmitter which generates PSD spikes around 60.1kHz. Each transmitter sends a single data frame at different times. We can clearly identify the data frames of the four transmitters in the frequency spectra. We use four 60Hz band-pass filters with center frequencies 60.13kHz, 67.09kHz, 67.3kHz, and 66.72kHz to extract the transmitted data frames. As shown in Appendix E.2, we achieve data exfiltration with a maximum bit error rate less than 7% for any transmitter.

6.2.3 Data exfiltration under different settings. Now, we vary the default settings listed in Section 6.1. The results are summarized in Table 1, while the figures are shown in Appendix E.1.

First, we run a concurrent program by playing “See You Again” on YouTube on a Google Chrome browser, which is one of the most viewed videos [62]. As video streaming can be fairly CPU intensive¹, running YouTube adds random variations to the transmitter’s power.

¹Our test desktop does not have a dedicated GPU to offload video processing.

Table 1: Summary of Data Exfiltration from Dell Optiplex

Scenario	Bit Error Rate	Bits Per Second	Detection Results
Default (4 cores)	0.0%	28.48	Fig. 27
With YouTube streaming	2.3%	27.82	Fig. 28(a)
With MS Word running	0%	28.48	Fig. 28(b)
With web browsing	0%	28.48	Fig. 28(c)
With HDD file transfer	3.5%	27.48	Fig. 28(d)
With ML training	1.67%	28.00	Fig. 28(e)
Loading 1 CPU core	8.9%	25.94	Fig. 29(a)
Loading 2 CPU cores	2.5%	27.77	Fig. 29(b)
Loading 3 CPU cores	0.0%	28.48	Fig. 29(c)
Using 4-bit pilot sequence	3.3%	28.13	Fig. 30(a)
Using 8-bit pilot sequence	0.0%	27.88	Fig. 30(b)

It also reduces the difference between transmitted “0”s and “1”s, resulting in a 2.3% bit error rate. We also run experiments using MS Word, web browsing, file transfer, and machine learning training as background applications resulting in 0%, 0%, 3.5%, and 1.67% bit error rates, respectively. In the MS word experiment, we mimic user behavior by repeatedly opening a new file, typing a few lines of texts and then saving the file. For the web browsing experiment, we open new popular websites (e.g., Gmail and Facebook), scroll through the page content, and follow links to other pages. For the file transfer experiment, we transfer a 5GB file from one HDD drive to another in our desktop computer running Windows 10. For the machine learning experiment, we repeat training Tensorflow in Python with 6000 samples from the MNIST data set taking around five minutes to finish [52]. From the results from Table 1 we see that under a diverse set of background applications running simultaneously, NoDE still maintains a low bit error rate.

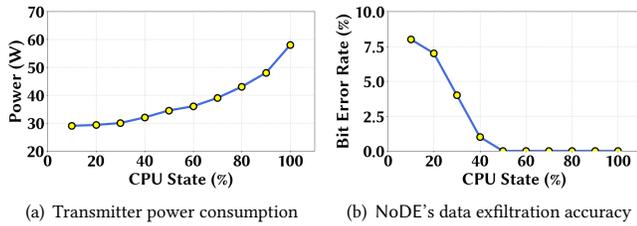


Figure 13: The impact of CPU power state on the bit error rate of NoDE and the transmitter’s (Dell Optiplex computer) power consumption. In our Dell Optiplex, 10% and 100% CPU states correspond to CPU frequencies of 0.79GHz and 3.68GHz, respectively.

Second, we use CPU pinning to restrict the number of cores that are assigned to the modulation program in NoDE. Reducing number of cores increases the bit error rate because it limits how much the modulation program can vary the transmitter’s current. Nonetheless, we find that even by loading only one CPU core (the weakest transmission), NoDE achieves an effective rate of 25.94 bits/second. In Table 1 we omit the four core case which is our default case with 0% bit error rate.

Finally, we consider 4-bit (“1101”) and 8-bit (“11001010”) pilot sequences. We see that when using a 4-bit pilot sequence “1101”, the bit detection threshold may not be properly set due to lack of enough pilot symbols, but it has lower overhead than the 8-bit pilot and NoDE still achieves 28.13 bits/second.

6.2.4 Impact of CPU scaling on the transmitter. We tune the CPU scaling of the transmitter by changing the “maximum processor state” in the Windows power management system. We vary the maximum processor state from 10% to 100% which, in our test computer, corresponds to CPU frequency from 0.79 GHz to 3.68 GHz. We use the default settings and transmit 100 frames under each different CPU scaling and record the average power consumption of the transmitter using a WattsUp power meter. Figs. 13(a) and 13(b) show the bit error rates and power consumption for the different CPU scaling. Similar to our experiment with the number of cores for NoDE, we see a higher bit error rate when the transmitter consumes less power due to reduced CPU speed. Note that, While dynamic CPU scaling at runtime is supported by modern CPUs, it is more commonly applied in data centers with sophisticated power-performance control, where energy saving is a crucial concern. For typical enterprise environments, DVFS is not applied and instead, desktop computers commonly run at the high-performance mode (i.e., 100% maximum processor state, which is also the default setting in Windows 10).

6.2.5 Data exfiltration without line of sight. As shown in Fig. 14, we run an experiment in Building B where the transmitter and receiver are plugged into two outlets located in two different rooms which are approximately 90 feet away from each other. We use a Dell PowerEdge computer as the transmitter which, albeit using a different switching frequency, adopts same PFC design as our Dell Optiplex computer. We show a snapshot of extracted signals at the receiver in Fig. 33 in Appendix E.4. We see that, compared to

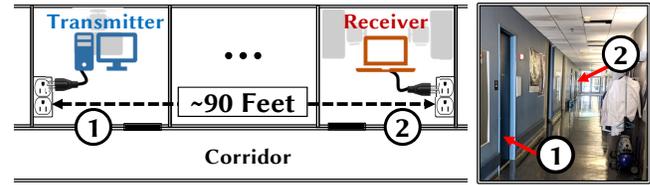


Figure 14: Experiment setup in Building B. ① Transmitter’s room. ② Receiver’s room. No line of sight between the transmitter and the receiver.

Fig. 27, the distinction between bit “1” and bit “0” in terms of the average amplitude of filtered voltage is less significant. Nonetheless, even with a transmitter-receiver distance of 90 feet and across different rooms, NoDE still achieves error-free stealthy data exfiltration with a rate of 28.48 bits/second.

6.2.6 Data exfiltration from other computers. To further evaluate NoDE, we run more experiments on four additional computers: an Apple iMac, an Acer, and two custom-built systems with top-brand power supply units (Corsair and EVGA). These computers are located in different labs in Building A. For the iMac, we could achieve a maximum of 15.79 bits/second due the iMac’s slow response to power change and wider frequency signature (discussed further in Appendix E.6). We summarize our results in Table 2, while the details are presented in Appendices E.5 and E.6.

In summary, while certain conditions may degrade the effectiveness of NoDE, our experiments on different computers and under different settings confirm that NoDE can exploit a building’s power network as a covert channel for stealthy data exfiltration from desktop computers without using a conventional communication network.

6.3 Discussions

We now discuss NoDE from the following important aspects.

Comparison with PLC adapters. Without dedicated physical powerline communications (PLC) adapters, NoDE still achieves information transfer over power networks. This is due to the PFC’s capability of generating prominent high-frequency quasi-orthogonal switching noises. During our experiments, we have found that the amplitudes of PFC-induced switching noise spikes and high-frequency voltage signals (in MHz range) modulated by a PLC adapter (NETGEAR PowerLINE 1000) are in the same order of magnitude. Thus, in practice, the achievable transmission range of NoDE is expected to be similar to that of a PLC adapter (typically up to a few tens of meters) [7].

Missed frames and bits. In any covert channels [29, 35, 73], some frames may not be successfully received due to erroneous pilot and/or payload symbols, and the receiver is not able to notify the transmitter due to the unidirectional covert channel. To trade efficiency for reliability, the transmitter may send each data frame multiple times and/or apply error correction coding [25].

Higher bit rate and limit. Like in the existing power-based data exfiltration literature [35, 73], we empirically demonstrate the achievable bit rate of NoDE. Nonetheless, we provide conjectures

Table 2: Summary of Experiments on Seven Different Computers.

Transmitting Computer	Configuration	Operating System	Power Supply Unit	Year	PFC Switching Frequency	Location	TX-RX Distance	Bit Error Rate	Bits Per Second	Detection Results
Dell Optiplex 9020	Core i7-4790, 16 GB	Windows 10	Dell-L290EM-01 300W by Lite-on Tech. Co.	2015	~67.3 kHz	Lab #1 (Building A)	~55 feet	0.0%	28.48	Fig. 27
Dell PowerEdge R630	Dual Xeon E52640, 32GB	Ubuntu Server 14.04	Dell-E495E-S1 495W by Astek Intl.	2016	~65.8 kHz	Office (Building B)	~90 feet	0.0%	28.48	Fig. 33
Dell XPS 8920	Core i7-7700, 16 GB	Windows 10	Dell-460AM-03 385W by Delta Electronics Inc.	2017	~60.1 kHz	Lab #1 (Building A)	~55 feet	0.0%	28.48	Fig. 31(d)
Acer G3-710	Core i7-7700, 16 GB	Ubuntu 16.04	ACER 750W	2016	~63.5 kHz	Lab #2 (Building A)	~20 feet	10.1%	25.60	Fig. 34(a)
Custom Built #1	Core i7-7700, 16GB	Windows 10	Corsair 850W RM850x-RPS0110	2018	~91.2 kHz	Lab #1 (Building A)	~55 feet	8.1%	26.17	Fig. 34(b)
Custom Built #2	Core i7-7700K, 16 GB	Ubuntu 16.04	EVGA 850W Supernova 850G2	2016	~67.7 kHz	Lab #3 (Building A)	~15 feet	9.2%	25.85	Fig. 34(c)
Apple iMac Model A1419 (27-inch)	Core i5-3470S, 8 GB	macOS 10.13.3	Apple 300W PA13112A1 (for 2012-2017 models)	2015	~101 kHz	Lab #1 (Building A)	~55 Feet	16% (50ms/sym) 2% (100ms/sym)	15.79 9.21	Fig. 35

on two possible approaches to further improving the achievable bit rate for data exfiltration. First, we may possibly improve the achievable bit rate if privileged access to the target computer’s PFC is granted. Specifically, the PFC’s feedback gain may be altered to increase the responsiveness of the power supply unit, i.e., the power supply unit follows changes in the CPU utilization/power consumption more closely. This can be achievable since most modern computer power supplies utilize digital control to adjust the feedback gain [64, 74]. Second, given privileged access to the computer’s power supply unit, another complementary approach is to modulate the PFC’s switching frequency for data exfiltration. The digitally-controlled PFC circuit allows dynamically setting the switching frequency through its control program [81]. However, this approach may still be restricted by how often we can change the switching frequency. Moreover, frequency modulation requires wider bandwidth and hence can be susceptible to greater interferences from other devices/computers.

Sources of bit errors. Various factors can introduce bit errors during data exfiltration, including the source signal strength (affected by the amount of transmitter’s power consumption that can be modulated by the malware and the PFC design), signal propagation path and fading (affected by the relative location/distance of transmitter and receiver, line impedance, building’s power network topology), interferences from other devices with similar PFC switching frequencies, among others. While it is challenging, if not impossible, to theoretically quantify the impact of different factors on the resulting bit errors for a given transmitter-receiver pair, one can *qualitatively* conclude based on standard bit error analysis for additive white-Gaussian noise channels [25] that a lower bit error rate can be achieved by increasing the source signal strength, reducing the signal propagation fading, and/or mitigating interferences. These are also reflected by our above empirical results.

Scalability of NoDE. The scalability of the simultaneous exfiltration depends largely on whether or not switching noise spikes generated by different computers overlap with each other. Next, we discuss the following three different cases – orthogonal switching

noises spikes, overlapping switching noise spikes, and practical scenarios.

- *When switching noise spikes of different computers are perfectly frequency-orthogonal.* In this case, data exfiltrations from different transmitting computers can be viewed as independent, without much inference from each other. If we conservatively assume that the switching noise spike of a computer occupies a frequency band of 500Hz (400Hz sidelobes around the most prominent spike plus 100Hz guard band) and that the noise spikes of all transmitting computers are perfectly frequency-orthogonal, then NoDE can achieve simultaneous data exfiltrations from *up to* 200 computers over the frequency range of 50–150kHz, which is the range for typical switching frequencies of PFC circuits [64].

- *When switching noise spikes of different computers are overlapping in frequencies.* In this case, simultaneous data exfiltrations become challenging, as in the case of any communications systems [25]. Thus, different transmitting computers need to access the covert channel at different times. This is not restrictive, since a target computer may not be always sending data over our covert channel.

- *Practical scenario.* In practice, the likelihood of overlapping PFC-induced noise spikes is not very high because the switching frequency is not tightly regulated in the power supplies. Thus, under a scale of up to a few tens of computers, we expect that the switching noise spikes of some computers may partially overlap, while most noise spikes do not overlap (Fig. 12). If other background computers have overlapping switching noise spikes with a target computer, their switching noises can be viewed as quasi-static background noises and do not significantly affect data exfiltrations from the target computer. For example, even on a single computer with simultaneous background applications, we show in Table 1 that NoDE still can successfully exfiltrate data.

To summarize, the likelihood of having overlapping PFC-induced noise spikes among two computers is not very high in practice, thus allowing simultaneous data exfiltrations. Nonetheless, when two or more target computers have the overlapping switching noise

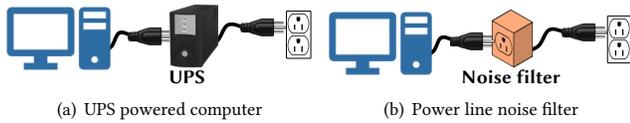


Figure 15: Possible defense mechanisms to prevent PFC switching noise from entering the power network. (a) Powering a computer through a UPS. (b) Adding power line noise filters.

spikes, they need to transmit information at different times using NoDE.

7 DEFENSE MECHANISM

Three major approaches exist to defend against NoDE— eliminating PFC-induced switching noises, preventing the switching noises entering the power network, and suppressing malware activities. The first two approaches involve hardware implementation and/or modification, while the last approach can be implemented primarily in software.

7.1 Eliminating PFC-induced Switching Noises

Completely eliminating PFC noise would require re-designing of computers’ power supply units with fundamentally different PFC strategy. However, it is non-trivial to find alternative solutions to replace the existing mature designs of power supply units without compromising energy efficiency. Moreover, such a change will require an industry-wide upgrade which is not likely to occur any-time soon. Alternatively, a stricter EMI regulation can be imposed to include components less than 150kHz.

7.2 Preventing Switching Noise From Entering Power Networks

An intuitive defense against NoDE is to power a computer through a UPS instead of directly connecting it to a power outlet, and thereby restricting the PFC noise from entering the power network. In such a case, as shown in Fig.15(a), the UPS sits between the desktop power supply and the power outlet (i.e., power network). However, an UPS does not necessarily provide electrical isolation from the power network. Instead, it “bypasses” the utility power to its connected devices during normal operation. The UPS acts as an alternate power source when the power supplied through the utility is interrupted in the event of voltage drop or complete power losses. To illustrate this, we connect our Dell PowerEdge computer to a 600VA CyberPower UPS and conduct our data exfiltration experiment in Building B. As shown in Fig. 36, we have “zero” bit error, which matches our previous experiments without the UPS. Hence, an UPS-powered computer does not necessarily mean it is immune to the threat of NoDE, let alone the added UPS cost.

Another defense is to insert a power line noise filters between computers and power outlets, as shown in Fig. 15(b). The filters are commercially available to use together with household/office appliances for reducing interference and better facilitating power line communications. However, they mainly reduce the amplitude of appliance-generated noise entering the power network, without

complete elimination. To demonstrate this, we run an experiment on our Dell PowerEdge computer plugged into a power line noise filter (X10 XPPF [78]) in Building B. The resulting received signal is shown in Fig. 37, from which we see that the signal amplitude of the PFC-induced switching noise spike is degraded by more than a factor of 10 compared to the case without any filters (Fig. 33). Thus, while not entirely prohibiting the transmitter’s switching noise spikes from entering the power network, the power line noise filter can significantly attenuate the amplitude of spikes, reducing the effective transmission distance.

7.3 Suppressing Malware Activities

As data exfiltration is done by varying the CPU load to modulate the transmitter’s overall power consumption, a possible defense is to randomly vary the CPU load to de-correlate the overall power consumption with information bits. In the prior literature, hardware-based techniques have been developed to randomize the power consumption and obfuscate the power signature of instructions executed in devices [3, 16, 27, 61]. While they are efficient in terms of the power overhead for randomization, such hardware-based techniques are typically tailored for devices with specific functions such as cryptography and not suitable for commodity processors. On the other hand, power randomization can also be achieved by software-based approaches.

Concretely, we evaluate software-based approaches for power randomization with two different implementations — RandomNoise and RandomPower. In RandomNoise, we design a program that launches CPU-intensive computations at random times to add random power consumption. In RandomPower, we follow the state-of-the-art technique to randomize the overall power consumption using a feedback loop [67]. As illustrated in Fig. 16, RandomPower samples the CPU power and uses combinations of CPU speed scaling (DVFS) and CPU-intensive computation to find-tune the power to follow the random pattern generator. The key difference between these two approaches is that RandomNoise mainly adds random power noise to the existing power consumption patterns, whereas RandomPower randomizes the overall power by more proactively controlling the CPU.

RandomNoise. We test NoDE’s performance under varying settings of RandomNoise’s time interval of added CPU loads, percent time of high CPU load, and the number of CPU cores used. Fig. 17(a) shows NoDE’s bit error rates as we increase the percent time of high CPU load from 10% to 90% for three different loading intervals (15/33/66 milliseconds). We see a general trend that, regardless of the interval length, an increasing percentage of CPU high load affects NoDE more. We also see that 15 milliseconds loading interval is worse than both 33 and 66 milliseconds intervals because it does not create a sustained high CPU load. On the other hand, we see from Fig. 17(b) that loading more CPU cores by RandomNoise also increases NoDE’s bit error rate.

RandomPower. We implement RandomPower in Ubuntu and use Intel RAPL interface for sampling the power [47, 65], `cpufreq` for DVFS [5], and repeated floating-point operations as CPU-intensive computation. While we implement RandomPower on Linux, software-based power monitoring and DVFS in other systems (e.g., Windows) are also available [5, 9, 10, 47]. In our experiment, the target power

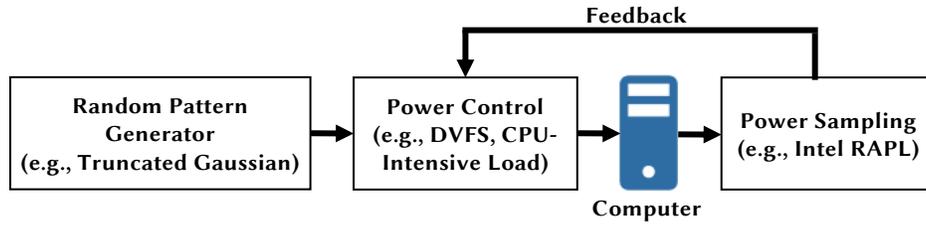


Figure 16: The building blocks of RandomPower which defends against NoDE by randomizing the computer’s power consumption.

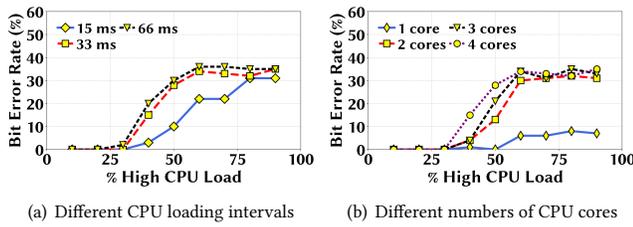


Figure 17: (a) and (b) Impact of RandomNoise on NoDE’s bit error rate for different settings.

is determined by random numbers generated following a Gaussian distribution where we discard values smaller than 0 and greater than 1, resulting in a truncated Gaussian distribution. The values of 0 and 1 corresponding to 35W and 85W in our experiment, respectively. The mean and variance are user-set inputs to RandomPower, and the resulting probability density function can be expressed as

$$f(x; \mu, \sigma) = \begin{cases} \frac{1}{\Phi(\mu, \sigma)\sigma\sqrt{2\pi}} e^{-\left(\frac{x-\mu}{\sigma}\right)^2}, & \text{if } 0 \leq x \leq 1, \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where $\Phi(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^1 e^{-\left(\frac{x-\mu}{\sigma}\right)^2} dx$, μ is the mean, and σ is the standard deviation of the untruncated Gaussian distribution. Based on the feedback, the CPU control block sets the appropriate CPU speed using DVFS and the amount of CPU-intensive workload to follow the randomly set power consumption target. In Fig. 18, we show a snapshot of the random number sequence and corresponding power consumption of RandomPower running with a mean of 0.5 and a standard deviation of 0.5 for a truncated Gaussian distribution. We update the randomly set target power every 40 milliseconds. We see that the power consumption closely follows the supplied random number with minor deviations at times mainly due to the computer power supply’s internal controls. While we use a truncated Gaussian distribution, this approach can be adopted with other probability distributions. In our evaluation, RandomPower works perfectly against NoDE, and we cannot even identify the pilot sequence to extract the transmitted bits.

Overhead. Both RandomNoise and RandomPower add overhead to the system to defend. Since RandomNoise injects random power noise, it results in additional power consumption by the computer. To have a detailed view of RandomNoise’s overhead, we show the power overhead under different cases in Fig. 19(a)

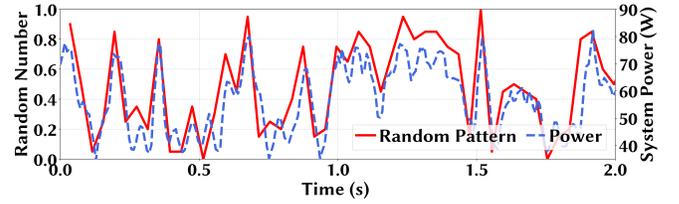


Figure 18: Illustration of the power variation following the target random number with a 0.5 mean and 0.5 standard deviation.

with the 15-millisecond results as outliers. We calculate the power overhead by running RandomNoise without any transmission and subtracting the idle power ($\sim 28W$) from the average power consumption. The key message from Fig. 19(a) is that RandomNoise can significantly affect NoDE’s performance ($> 30\%$ bit error rate) when it injects more than 20W of random power consumption. Nonetheless, in relation to the 28W idle power of our test computer, this amounts to a 70% overhead.

For RandomPower, we have both power overhead when CPU-intensive workloads are launched to increase the overall power, and performance overhead when CPU speed is throttled to reduce the overall power. To evaluate the power and performance overheads, we run RandomPower with four different applications — Machine learning training (ML), large file transfer in hard disk (IO), word processing (Word), and web browsing (Web). Figs. 19(b) and 19(c) show the change in power and performance overheads as we change the average power consumption target by changing the mean of the truncated Gaussian random distribution. We calculate the power overhead by subtracting the average power consumption without RandomPower from the average power consumption with RandomPower. For the performance overhead, we take the ML training and file transfer completion times with RandomPower and normalize the values with respect to those without RandomPower. We exclude Word and Web from the performance overhead evaluation since they do not have completion times like ML or IO applications. Note that during our experiments, we do not experience any significant perceivable impact of RandomPower on the performance of word processing and web browsing.

We see that the power overheads increase with increasing average power consumption. Also, RandomPower’s power overhead with perfect defense is lower than RandomNoise’s power overhead at higher bit error rates. More importantly, however, we see

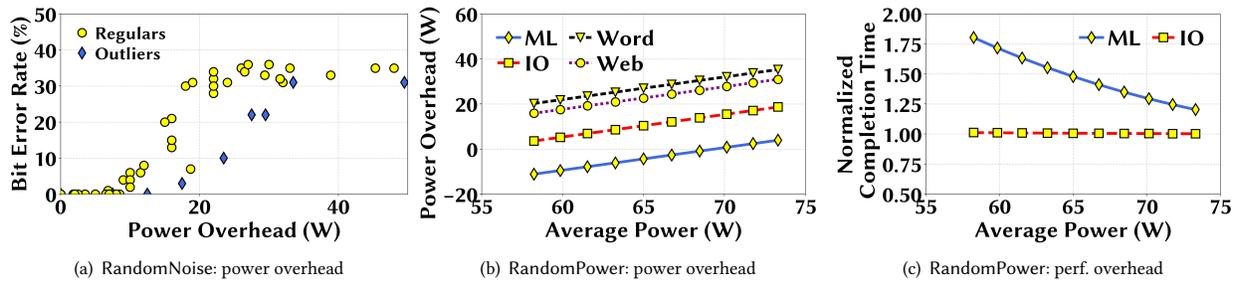


Figure 19: (a) The power overhead of RandomNoise at different levels of effectiveness (i.e., NoDE’s bit error rates). The 15 ms CPU load interval setting is marked as outliers. (b) Power overhead of RandomPower for different target average power. (c) Performance overhead of RandomPower. Completion time is normalized to the case without RandomPower.

that different applications have different power overheads for the same average power, with ML having a negative power overhead when the average power is less than 70W. This is due to each application’s power requirement. RandomPower running along an application with a low power requirement mostly adds power to follow its power target, whereas RandomPower needs to apply CPU throttling more frequently to reduce power when the underlying application requires more power. This is also why ML has a negative power overhead, indicating that due to frequent throttling RandomPower has reduced the average power lower than the otherwise average power requirement of ML without RandomPower. From Fig. 19(c), it is also evident that frequent throttling at lower average power targets causes a higher performance overhead for the ML application. For the IO application, we do not see a significant performance variation since CPU throttling does not severely affect the file transfer speed. The key take-away from the results is that the overhead for RandomPower depends on the underlying application and, for power-hungry applications, reducing the power overhead comes at the expense of performance degradation. A favorable balance between power and performance overheads can be attained through careful choice of the random number distribution parameters (e.g., mean).

RandomNoise vs RandomPower. RandomPower is more effective against NoDE than RandomNoise, since RandomPower offers a perfect defense against NoDE and incurs a lower power overhead as well, especially for power-hungry background applications. However, RandomPower incurs possible performance overheads due to the CPU throttling. It may also require additional OS privileges and/or accesses for CPU throttling and instrumentation for power sampling.

In addition to the aforementioned pro-active countermeasures, a reactive defense approach would be to identify and remove data exfiltration malware by monitoring application behavior or computer CPU utilization. However, the constant emergence of new malware remains as, and will continue being, one of the greatest threats faced by computers [4, 59, 70, 72]. On the other hand, utilizing the knowledge of NoDE for data exfiltration, one can design a power network voltage monitoring system that continuously scans through the voltage signals for suspicious switching noise spike patterns in the high frequency. A potential drawback of this approach is the computational burden to continuously monitor a

large frequency spectrum since PFC-induced switching noise is generated by every computer power supply unit.

In summary, we see that different countermeasures against NoDE have their own merits and hurdles. Based on our study, as a hardware-based defense, we recommend the installation of power noise filters because of its attenuation on PFC-induced switching noises. As for the software-based technique, we recommend power randomization due to its effectiveness.

8 RELATED WORK

There have been a plethora of studies on data exfiltration under a threat model where an adversary tries to extract information from a tightly secured computer system without using traditional data transfer protocols (e.g., network). The key idea is to encode information in *physical* attributes (e.g., the heat generated by a computer [31]) to carry it to an external receiver (e.g., temperature sensor). Meanwhile, decoding changes of these physical attributes does not require any cyber access to the target system, thus bypasses the system’s defense and forms covert channels for stealing information. Alternatively, a secure system may spill its secrets by inadvertently influencing an externally visible physical property (i.e., a side channel) [22–24].

Table 3 summarizes the physical medium, key design attributes, transmission rates, and effective distances of the recently proposed data exfiltration attacks. Compared to the existing research, NoDE achieves a reasonably high bit rate of 28.48 bits/s. Most acoustic covert/side channels cannot achieve a transfer rate higher than NoDE, except for [8] (140 bits/s) which requires the target to be equipped with a speaker and the receiver be in the same room as the target computer. [8] can reach up to 67,000 bits/s only in the audible range (20Hz~20kHz) at the expense of high detection possibility. On the other hand, both electromagnetic emanation and magnetic covert/side channel can achieve similar transfer rates as NoDE but have much shorter effective distances. [29] can reach a commendable speed of 1000 bits/s but requires professional-grade receiver hardware with a high sophistication.

Among all, optical covert channels attain the best transmission rates because of the extremely fast response time of LEDs. But, they require the receiver to be in the line of sight of the transmitter (e.g., be in the same room). Also, because of the high bit rates, photodiodes need to be used as the receiver, further restricting the

Table 3: Summary of Data Exfiltration Attacks with Different Media.

Medium	Proposed Design	Bit Rate	Effective Distance
Acoustic	HDD noise [32], Fan noise [34], Mesh network [38], RSA key extraction [24], Computer speakers [8] Gyroscope modulation using ultrasound [21]	0.25~140 bits/s	0~11 meters
Thermal	Computer generated heat [31]	0.0022 bits/s	0.4 meters
Electromagnetic emanation	CRT monitor EMI radiation[48], Memory bus [29], Extracting cryptographic key [6, 22, 23]	2~512 bits/s	0~20 meters
Magnetic	Hard drive head [58], Escaping Faraday cage [36]	4~40 bits/s	0.15~1.5 meters
Optical	Equipment status LED [55], HDD LED [37]	4~56 kbits/s	Line of sight
Power	Power consumption [35], Mobile's charging power [73], Key extraction from mobile's power analysis [23]	2~1000 bits/s (projected)	Length of power/charging cable

effective distance even within the line of sight. In contrast, NoDE can have the transmitter and receiver in two different rooms that are 27.4 meters away from each other without line of sight.

Another important aspect of NoDE is that unlike other studies which cannot achieve both their highest bit rate and longest distance for the same settings (i.e., increasing distance decreases bit rate), NoDE works at the 27.4 meters distance without compromising its rate of 28.48 bits/s at 0% bit error. While Table 3 is not an exhaustive list, it provides important insights into the potential and limitation of various physical covert/side channels-based data exfiltration attacks.

9 CONCLUDING REMARKS

In this paper, we studied data exfiltration from a desktop computer in an enterprise environment, and proposed NoDE to achieve stealthy information transfer over a building's power network without using any PLC adapters. NoDE exploits high-frequency switching noises caused by the PFC circuits built into all of today's computers and achieves an effective rate of 28.48 bits/second with a distance of 90 feet (27.4 meters) without line of sight. We validated NoDE's data exfiltration capability under different settings and hardware configurations. We also showed that certain configurations such as CPU speed scaling may reduce NoDE's data exfiltration rates. In addition, we offered some insights into the limitations and open issues of our proposed system. Finally, we outlined a few possible defenses and suggested both hardware-based and software-based techniques.

REFERENCES

- [1] Ross Anderson and Markus G. Kuhn. Soft tempest - an opportunity for nato, 1999.
- [2] Ben Hunter. L0rdix: Multipurpose attack tool, <https://blog.ensilo.com/l0rdix-attack-tool>.
- [3] Luca Benini, Elvira Omerbegovic, A Macii, Massimo Poncino, E Macii, and Fabrizio Pro. Energy-aware design techniques for differential power analysis protection. In *Proceedings 2003. Design Automation Conference (IEEE Cat. No. 03CH37451)*, pages 36–41. IEEE, 2003.
- [4] Ralf Benzmueller. Malware trends. *G Data Security Blog*, 2017 (<https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017>).
- [5] Dominik Brodowski. Cpu frequency and voltage scaling code in the linux(tm) kernel, <https://www.kernel.org/doc/Documentation/cpu-freq/user-guide.txt>.
- [6] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. Screaming channels: When electromagnetic side channels meet radio transceivers. In *CCS*, 2018.
- [7] Cristina Cano, Alberto Pittolo, David Malone, Lutz Lampe, Andrea M. Tonello, and Anand G. Dabak. State of the art in power line communications: From the applications to the medium. *IEEE J. Sel. A. Commun.*, 34(7):1935–1952, July 2016.
- [8] Brent Carrara and Carlisle Adams. On acoustic covert channels between air-gapped systems. In *Foundations and Practice of Security*, 2014.
- [9] Hardware Dev Center. Powercfg command-line options, <https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/powercfg-command-line-options>.
- [10] AMD Developer Central. AMD uProf, <https://developer.amd.com/amd-uprof/>.
- [11] Swarup Chandra, Zhiqiang Lin, Ashish Kundu, and Latifur Khan. Towards a systematic study of the covert channel attacks in smartphones. In *SecureComm*, 2014.
- [12] Long Cheng, Fang Liu, and Danfeng Yao. Enterprise data breach: Causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(5), 2017.
- [13] Shane S Clark, Hossen Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. Current events: Identifying webpages by tapping the electrical outlet. In *ESORICS*, 2013.
- [14] Shane S Clark, Hossen Mustafa, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, Kevin Fu, A Rahmati, M Salajegheh, D Holcomb, et al. Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *HealthTech*, 2013.
- [15] Gabe Cohn, Daniel Morris, Shwetak N. Patel, and Desney S. Tan. Your noise is my command: Sensing gestures using the body as an antenna. In *CHI*, 2011.
- [16] Debayan Das, Shovan Maity, Saad Bin Nasir, Santosh Ghosh, Arijit Raychowdhury, and Shreyas Sen. High efficiency power side-channel attack immunity using noise injection in attenuated signature domain. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67. IEEE, 2017.
- [17] Electronic Code of U.S. Federal Regulations. Unintentional radiators, section 15.107 – conducted limits, 2018.
- [18] Energy Star. Computers specification version 7.0, 2018, https://www.energystar.gov/products/spec/computers_specification_version_7_0_pd.
- [19] Miro Enev, Sidhant Gupta, Tadayoshi Kohno, and Shwetak N. Patel. Televisions, video privacy, and powerline electromagnetic interference. In *CCS*, 2011.
- [20] Miro Enev, Sidhant Gupta, Tadayoshi Kohno, and Shwetak N. Patel. Televisions, video privacy, and powerline electromagnetic interference. In *CCS*, 2011.
- [21] Benyamin Farshteindiker, Nir Hasidim, Asaf Grosz, and Yossi Oren. How to phone home with someone else's phone: Information exfiltration using intentional sound noise on gyroscopic sensors. In *WOOT*, Berkeley, CA, USA, 2016.
- [22] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *CHES*, 2015.
- [23] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. In *CCS*. ACM, 2016.
- [24] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In *Crypto*, 2014.
- [25] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.
- [26] Cassi Goodman. An introduction to tempest, 2001, <https://www.sans.org/reading-room/whitepapers/privacy/paper/981>.
- [27] Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 33–48. Springer, 2011.
- [28] Mordechai Guri and Yuval Elovici. Bridgeware: The air-gap malware. *Commun. ACM*, 61(4):74–82, March 2018.
- [29] Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, and Yuval Elovici. Gsmem: Data exfiltration from air-gapped computers over GSM frequencies. In *USENIX Security*, 2015.
- [30] Mordechai Guri, Matan Monitz, and Yuval Elovici. Bridging the air gap between isolated networks and mobile phones in a practical cyber-attack. *ACM Trans.*

- Intell. Syst. Technol.*, 8(4):50:1–50:25, May 2017.
- [31] Mordechai Guri, Matan Monitz, Yisroel Mirski, and Yuval Elovici. BitWhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *CSF*, 2015.
- [32] Mordechai Guri, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici. Acoustic data exfiltration from speakerless air-gapped computers via covert hard-drive noise ('diskfiltration'). In *ESORICS*, 2017.
- [33] Mordechai Guri, Yosef Solewicz, Andrey Daidakulov, and Yuval Elovici. MOSQUITO: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication. <https://arxiv.org/abs/1803.03422>, 2018.
- [34] Mordechai Guri, Yosef A. Solewicz, Andrey Daidakulov, and Yuval Elovici. Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers. <http://arxiv.org/abs/1606.05915>, 2016.
- [35] Mordechai Guri, Boris Zadov, Dima Bykhovskiy, and Yuval Elovici. Power-Hammer: Exfiltrating data from air-gapped computers through power lines. <https://arxiv.org/abs/1804.04014>, 2018.
- [36] Mordechai Guri, Boris Zadov, Andrey Daidakulov, and Yuval Elovici. ODINI: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields. <https://arxiv.org/abs/1802.02700>, 2018.
- [37] Mordechai Guri, Boris Zadov, and Yuval Elovici. LED-it-GO: Leaking (a lot of) data from air-gapped computers via the (small) hard drive led. In *DIMVA*, 2017.
- [38] Michael Hanspach and Michael Goetz. On covert acoustical mesh networks in air. *Journal of Communications*, 8(11):758–767, November 2013.
- [39] Timothy Hegarty. An overview of radiated EMI specifications for power supplies. *Texas Instruments Whitepaper*, June 2018.
- [40] Arik Hesseldahl. The tempest surrounding tempest, 2000, <https://www.forbes.com/2000/08/10/mu9.html#725665731004>.
- [41] Harold Joseph Highland. Random bits & bytes: Electromagnetic radiation revisited. *Computers and Security*, 1986.
- [42] IDC. PC market (2nd quarter of 2018), <https://www.idc.com/getdoc.jsp?containerId=prUS44118818>.
- [43] IEC. IEC 61000-3-2:2018: Electromagnetic compatibility (EMC) - part 3-2, <https://www.youtube.com/watch?v=F8nL2gRYTc0&t=3s>.
- [44] Mohammad A. Islam and Shaolei Ren. Ohm's law in data centers: A voltage side channel for timing power attacks. In *CCS*, 2018.
- [45] Kaspersky Lab. Shadowpad: How attackers hide backdoor in software used by hundreds of large companies around the world. *Corporate News*, August 2017.
- [46] Kevin Jones. Take a look at l0rdix, the super malware toolkit of 2018, <https://hackercombat.com/take-a-look-at-l0rdix-the-super-malware-toolkit-of-2018>.
- [47] Patrick Konsor. Intel power gadget, <https://software.intel.com/en-us/articles/intel-power-gadget>.
- [48] Markus G Kuhn and Ross J Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *International Workshop on Information Hiding*, pages 124–142. Springer, 1998.
- [49] David Kushner. The real story of stuxnet, <https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet>.
- [50] Butler W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10):613–615, October 1973.
- [51] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.
- [52] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [53] Pavel Lifshits, Roni Forte, Yedid Hoshen, Matt Halpern, Manuel Philipose, Mohit Tiwari, and Mark Silberstein. Power to peep-all: Inference attacks by malicious batteries on mobile devices. In *Privacy Enhancing Technologies*, 2018.
- [54] Yannan Liu, Lingxiao Wei, Zhe Zhou, Kehuan Zhang, Wenyuan Xu, and Qiang Xu. On code execution tracking via power side-channel. In *CCS*, 2016.
- [55] Joe Loughry and David A. Umphress. Information leakage from optical emanations. *ACM Trans. Inf. Syst. Secur.*, 5(3):262–289, August 2002.
- [56] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.
- [57] Claudio Marforio, Hubert Ritzdorf, Aurélien Francillon, and Srdjan Capkun. Analysis of the communication between colluding applications on modern smartphones. In *ACSAC*, 2012.
- [58] Nikolay Matyunin, Jakub Szefer, Sebastian Biedermann, and Stefan Katzenbeisser. Covert channels using mobile device's magnetic field sensors. In *ASP-DAC*, 2016.
- [59] McAfee Labs. Threats report, March 2018.
- [60] David Meisner and Thomas F. Wenisch. Peak power modeling for data center servers with switched-mode power supplies. In *ISLPED*, 2010.
- [61] Amir Moradi and Alexander Wild. Assessment of hiding the higher-order leakages in hardware. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 453–474. Springer, 2015.
- [62] Official Video on YouTube. See you again, <https://www.youtube.com/watch?v=RgKAFK5djSk>.
- [63] On Semiconductor. New evidence of hacked supermicro hardware found in u.s. telecom, <https://www.bloomberg.com/news/articles/2018-10-09/new-evidence-of-hacked-supermicro-hardware-found-in-u-s-telecom>.
- [64] On Semiconductor. Power factor correction (PFC) handbook, <http://www.onsemi.com/pub/Collateral/HBD853-D.PDF>.
- [65] Srinivas Pandravadu. Running average power limit, <https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl>.
- [66] J. Park, X. Xu, Y. Jin, D. Forte, and M. Tehranipoor. Power-based side-channel instruction-level disassembler. In *DAC*, 2018.
- [67] Raghavendra Pradyumna Pothukuchi, Sweta Yamini Pothukuchi, Petros Voulgaris, and Josep Torrellas. Maya: Falsifying power sidechannels with operating system support. *arXiv preprint arXiv:1907.09440*, 2019.
- [68] David E. Sanger and Thom Shanker. NSA devises radio pathway into computers. In *The New York Times*, Jan. 14, 2014.
- [69] SANS Institute. Covert channels. *InfoSec Reading Room*, 2010.
- [70] SANS Institute. Threat landscape survey: Users on the front line. *InfoSec Reading Room*, 2017.
- [71] Security Stronghold. How to remove senna spy ftp trojan, <https://www.securitystronghold.com/gates/senna-spy-ftp-trojan.html>.
- [72] SophosLabs. Malware forecast. *Sophos White Paper*, 2018.
- [73] Riccardo Spolaor, Laila Abudahi, Veelasha Moonsamy, Mauro Conti, and Radha Poovendran. No free charge theorem: A covert channel via usb charging cable on mobile devices. In *ACNS*, 2017.
- [74] STMicroelectronics. Circuits for power factor correction with regards to mains filtering, <http://www.onsemi.com/pub/Collateral/HBD853-D.PDF>.
- [75] Symantec. Sennaspy generator, <https://www.symantec.com/security-center/writeup/2001-062211-2540-99>.
- [76] Mohammad Tehranipoor and Farinaz Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design & Test*, 27(1):10–25, January 2010.
- [77] Wikipedia. Data theft, https://en.wikipedia.org/wiki/Data_theft.
- [78] X10 Home Gadgets. Xppf plug in noise filter, <https://www.x10.com/xppf-plugin-filter.html>.
- [79] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, and Kiran S. Balagani. On inferring browsing activity on smartphones via USB power analysis side-channel. *IEEE Transactions on Information Forensics and Security*, 12(5):1056–1066, May 2017.
- [80] Jindong Zhang, Jianwen Shao, Peng Xu, Fred C. Lee, and Milan M. Jovanovic. Evaluation of input current in the critical mode boost pfc converter for distributed power systems. In *IEEE Applied Power Electronics Conference and Exposition*, 2001.
- [81] Wanfeng Zhang, Guang Feng, Yan-Fei Liu, and Bin Wu. A digital power factor correction (pfc) control strategy optimized for dsp. *IEEE Transactions on Power Electronics*, 19(6):1474–1485, 2004.

APPENDIX

A RELATED NOTIONS

Power spectrum/ frequency spectrum: Power spectrum disintegrates a signal into its frequency components and show the power of each frequency component. To illustrate the changes in happening in different frequency components of a signal, the power spectrum or frequency spectrum is typically shown over time with heat maps.

Passband: In signal processing, filters are applied on a signal to attenuate undesired frequency components while let *pass* the useful frequency components. The passband refers to the frequency range that a filter allows to pass through. The passband is identified using the lower and upper cutoff frequencies. When a filter allows a specific frequency band to pass, it is called a band-pass filter. When the passband starts at zero (i.e., lower cutoff = 0 Hz), it is called a low-pass filter. When the passband ends at infinity (i.e., upper cutoff - infinity Hz) it is called a high-pass filter.

Harmonics: Harmonics are frequency components at multiples of the fundamental frequency. In power system, harmonics are produced when the 50Hz/60Hz sinusoidal voltage or current gets distorted by non-linear loads (such as SMPS). Harmonics in the power system create unwanted losses in power transmission.

B VOLTAGE FROM A POWER OUTLET

We show in Fig. 20(a) a snapshot of the voltage trace collected from a power outlet in our lab. It can be seen that the supplied voltage varies by more than 500mV within just a few minutes. We further show the probability mass function (PMF) of a 24-hour voltage distribution in Fig. 20(b), demonstrating a nearly 5V variation in the actual supplied voltage.

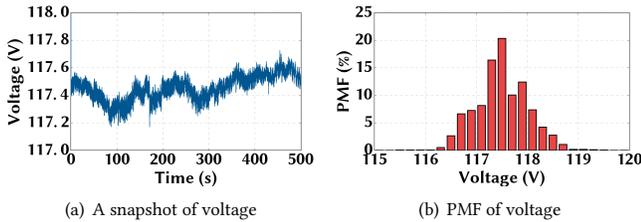


Figure 20: The voltage of a power outlet varies over time.

C COMPUTER’S POWER SUPPLY UNIT

C.1 Conduction Modes for PFC

In Table 4, we show a summary of major conduction modes for PFC circuits used in today’s computers.

C.2 PFC Switching Frequency Variation

We show the cumulative density function (CDF) of our Dell Optiplex’s PFC switching frequency variation in Fig. 21(a). We see that the PFC switching frequency varies no more than 50Hz within 5 seconds. In Fig. 21(b), we also show the frequency spectrum over a 5 second window and confirm that there is only a small variation in the PFC switching frequency.

Table 4: Summary of Major Conduction Modes for PFC [64]

Conduction Mode	Power Rating	Current Waveform	Frequency Analysis	Property
Continuous Conduction Mode (CCM)	>300W			- Fixed frequency - Large inductor - Lowest peak current
Discontinuous Conduction Mode (DCM)	<300W			- High peak current - Reduced inductance - Good stability
Critical Conduction Mode (CrCM)	<300W			- Varying frequencies - High peak current

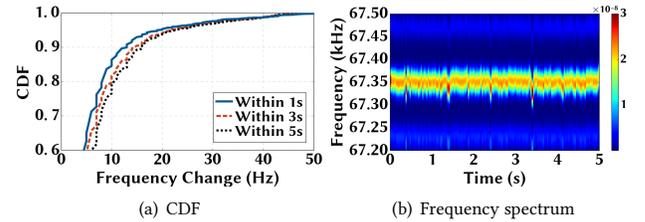


Figure 21: PFC switching frequency variation over time.

C.3 Current Drawn by Microsoft Surface Book

We show the measured current drawn by a Microsoft Surface Book with a 65W power rating in Fig. 22(a) and the frequency analysis of the current in Fig. 22(b). We see that the current waveform looks by no means like the sine voltage waveform and the total harmonic components are even stronger than the 60Hz component in terms of power spectral density (PSD). Beyond 20kHz, the intensity of harmonics decreases to an extremely low level, leaving only background noise.

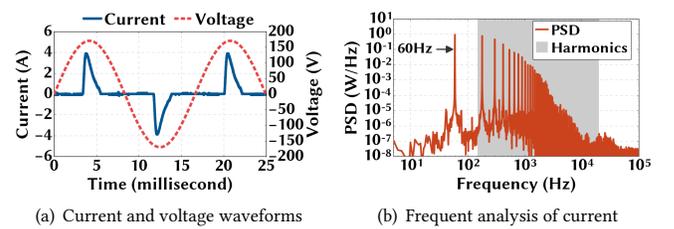


Figure 22: The current drawn by a Microsoft Surface Book contains a significant amount of harmonics.

D ALGORITHM FOR CURRENT MODULATOR

We design a simple current modulator as described in Algorithm 3. The current modulator program takes 1/0 bit streams as the input and runs some dummy calculations (e.g., generating random numbers) to load the CPU and change the computer’s input current.

Algorithm 3 Current Modulator

```

1: Input: Bit stream  $B$  and symbol duration  $T$ 
2: for every bit  $B_i$  do
3:   if  $B_i == 1$  then
4:     Run dummy calculations for  $T$  milliseconds
5:   else
6:     Idle for  $T$  milliseconds
7:   end if
8: end for

```

E DATA EXFILTRATION RESULTS

E.1 Experiment on Dell Optiplex Computer

NoDE with different background application. Now, we run experiments on our Dell Optiplex computer under settings different from the default one. First, to run a concurrent program to mimic user’s normal activity, we play “See You Again” on YouTube on a Google Chrome browser, which is one of the most viewed videos [62]. We also run experiments using MS Word, web browsing, file transfer, and machine learning training as background applications resulting in 0%, 0%, 3.5%, and 1.67% bit error rates, respectively. In the MS word experiment, we mimic user behavior by repeatedly opening a new file, typing a few lines of texts and then saving the file. For the web browsing experiment, we open new popular websites (e.g., Gmail and Facebook), scroll through the page content, and follow links to other pages. For the file transfer experiment, we transfer a 5GB file from one HDD drive to another in our desktop computer running Windows 10. For the machine learning experiment, we repeat training Tensorflow in Python with 6000 samples from the MNIST data set taking around five minutes to finish [52]. We show snapshots of the detections in Fig. 28.

NoDE with different number of CPU cores. Next, we restrict the number of cores that are assigned to the modulation program in NoDE, and show the experimental results in Fig. 29.

NoDE with different pilot lengths. Finally, we consider 4-bit (“1101”) and 8-bit (“11001010”) pilot sequences, and show the experiment results in Fig. 30.

E.2 Experiment Using Multiple Transmitters

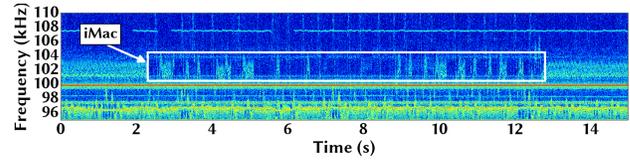
Fig. 31 shows the snapshot of data exfiltration for different transmitters from our multi-transmitter experiment. Our results show 0% error for TX#1 and TX#4 while 6.8% and 1.1% error for TX#2 and TX#3, respectively.

E.3 Impact of CPU scaling on the transmitter

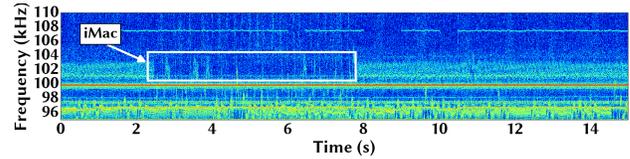
Snapshots of detection results for 10%, 50%, and 100% CPU states are shown in Fig. 32.

E.4 Experiment Without Line of Sight

Fig. 33 shows the snapshot of data exfiltration when the receiver and transmitters are placed in two separate rooms 90 feet away from each other in Building B.



(a) Bit length = 100ms



(b) Bit length = 50ms

Figure 23: Frequency spectrum showing a data frame transmitted by the iMac.

E.5 Experiment on Other Computers

Fig. 34 shows the experiment results for our Acer computer, our custom built computer #1, and our custom built computer #2.

E.6 Experiment on iMac

We conduct our experiment on an Apple iMac 27-inch computer in Lab #1. The transmitter and the receiver are placed 55 feet away from each other. The iMac has an all-in-one compact design with monitor and other components assembled together and powered by a single power supply unit.

We first show the frequency spectrum for a data frame transmitted from the iMac with a symbol length of 100ms in Fig. 23(a), which reveals that the PFC-induced frequency spike of the iMac is around 101kHz. It also shows that frequency band that carries the transmitted data has a much wider bandwidth of \sim 1kHz compared to the Dell computers’ frequency bands in Fig. 12. Further, while not explicitly indicated in Fig. 23(a), the frequency amplitudes in the transmitted band is reversed, where the “1”s has lower amplitude and “0”s has a higher amplitude. We also show the frequency spectrum for a symbol length of 50ms in Fig. 23(b). The figures show the detection accuracy significantly deteriorates as we increase the symbol rate. In fact, we can not have successful data transmission for our default symbol length of 33ms. This is mainly due to the iMac’s power supply unit has a slower response to the power demand compared to other computers in our experiments.

Next, we show a snapshot of data exfiltration from the iMac for four different symbol lengths ranging from 50ms to 125ms in Fig. 35. As discussed before, the bit error rate decreases as we increase the symbol length. However, since a higher symbol length means a lower maximum bits/second, the overall bit transfer rate decreases.

F PMF OF NOISE IN EXTRACTED SIGNAL AMPLITUDES

We set the average signal amplitude of filtered voltage signals when transmitted bits are 1 as the reference signal amplitude for bit 1. When the transmitted bit is 1, any deviation of an actually received bit-wise signal amplitude from the reference value is considered

as noise. Similarly, we also obtain the reference signal amplitude for bit 0, and obtain the noise. Next, we show the PMF of noise for both Dell Optiplex and Dell PowerEdge computers in Fig. 24. The noise amplitude distribution does not seem to be Gaussian. In other words, our covert channel is likely corrupted by non-Gaussian noises.

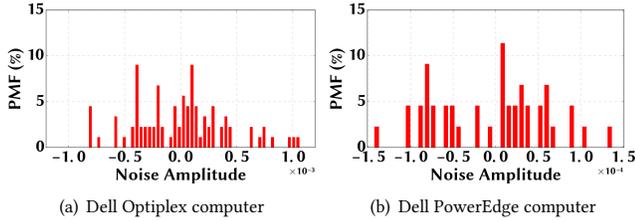


Figure 24: PMF of noise in extracted signal amplitudes.

G DEFENSE MECHANISM EXPERIMENTS

G.1 Experiment on a UPS-Powered Computer

We power our Dell PowerEdge computer through a CyberPower UPS, which is plugged into a power outlet in the transmitter’s room in Building B. In Fig. 36, we show that the receiver can still extract information without errors. Thus, an UPS-powered computer does not necessarily mean it is immune to the threat of NoDE, let alone its added UPS cost.

G.2 Experiment in the Presence of Noise Filters

We plug in noise filter (X10 XPPF) [78] into a power outlet in the transmitter’s room in Building B, and then plug in the power cord of our PowerEdge computer into the noise filter. Although the intensity of high-frequency PSD spikes in the receiver’s voltage signal is reduced, it is still much higher than the power line background noise and detectable. In Fig. 37, we show that the receiver can have an effective rate of 25.57 bits/second. Although the transmission distance is reduced compared to the no-filter case, the receiver can still be located in another room without being restricted to the line of sight of the transmitter.

G.3 Random Power Load Defense

We test the performance of a defense program that that injects random power loads by randomly deciding to either run a CPU-intensive computation or remain idle. We test NoDE’s performance under varying settings of the defense program’s time interval of the CPU loads, percent time of high CPU load, and number of CPU cores used by the defense program. Figs. 38, 39, 40, and 41 shows the snapshot of detection results for some selected cases of interest, while Fig. 25 shows the power overhead for different settings. Fig. 26 shows the current drawn by the defense program with the CPU load 20% and 60% of the times.

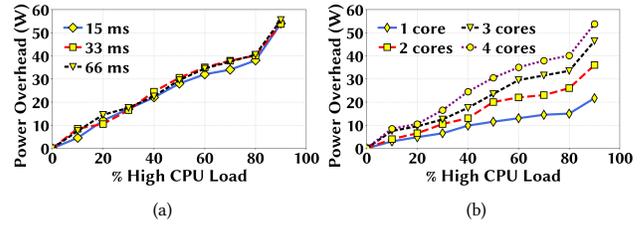


Figure 25: Power overhead for the defense program. (a) Power overhead remains same for different CPU load intervals. (b) Impact of the number of cores utilized by the defense program.

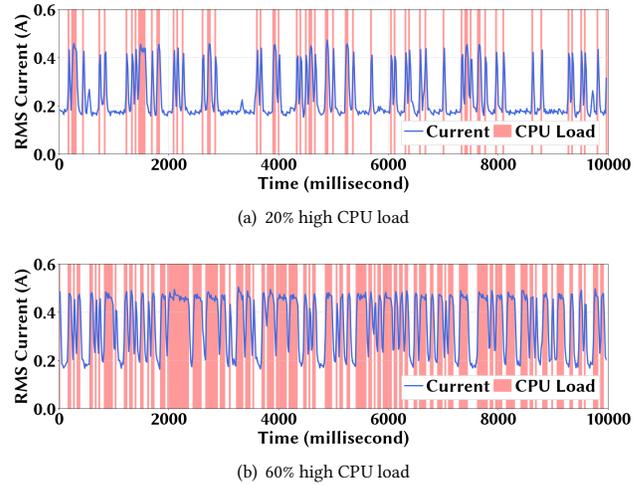


Figure 26: Current draw of the defense program running with random high CPU loads. (a) 20% high CPU load. (b) 60% high CPU load.

H SNAPSHOTS OF ALL DETECTION RESULTS

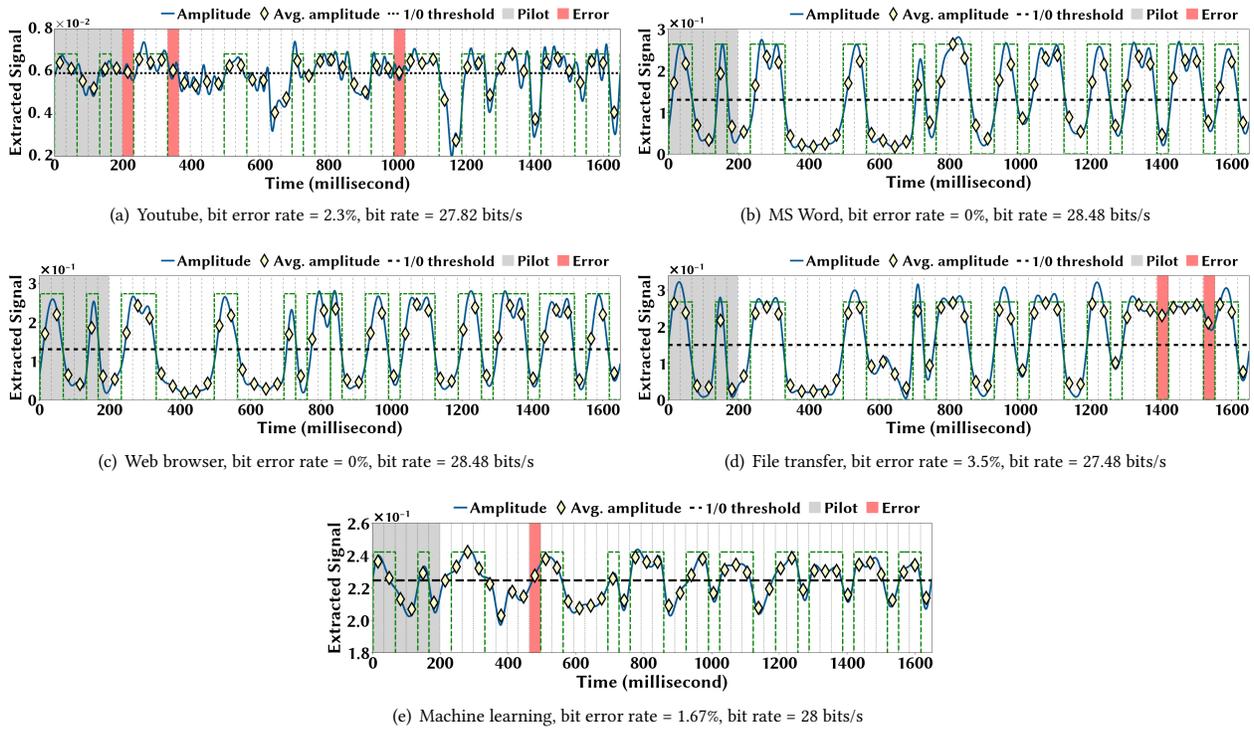


Figure 27: Different background applications.

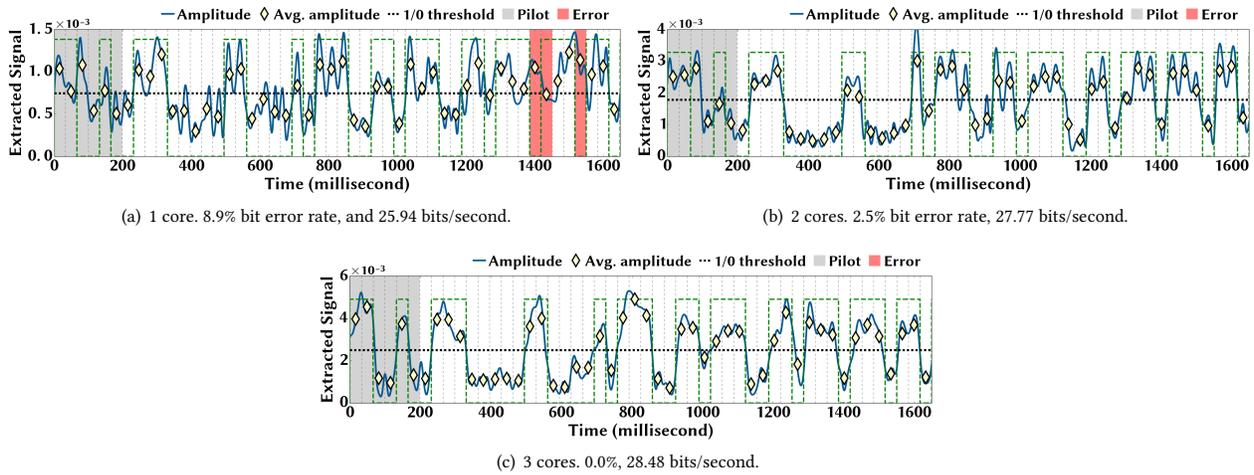


Figure 28: Dell Optiplex computer with different numbers of CPU cores assigned to the modulation program in NoDE. The receiver filters its received voltage signals with passband of $\langle 67.28\text{kHz}, 67.34\text{kHz} \rangle$.

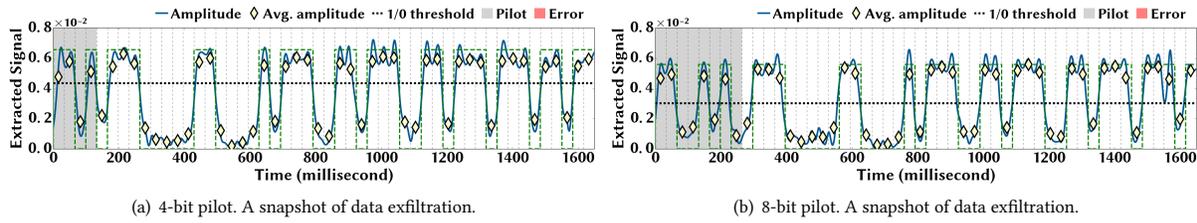


Figure 29: Dell Optiplex computer with different pilot sequences. The receiver filters its received voltage signals with passband of $\langle 67.28\text{kHz}, 67.34\text{kHz} \rangle$.

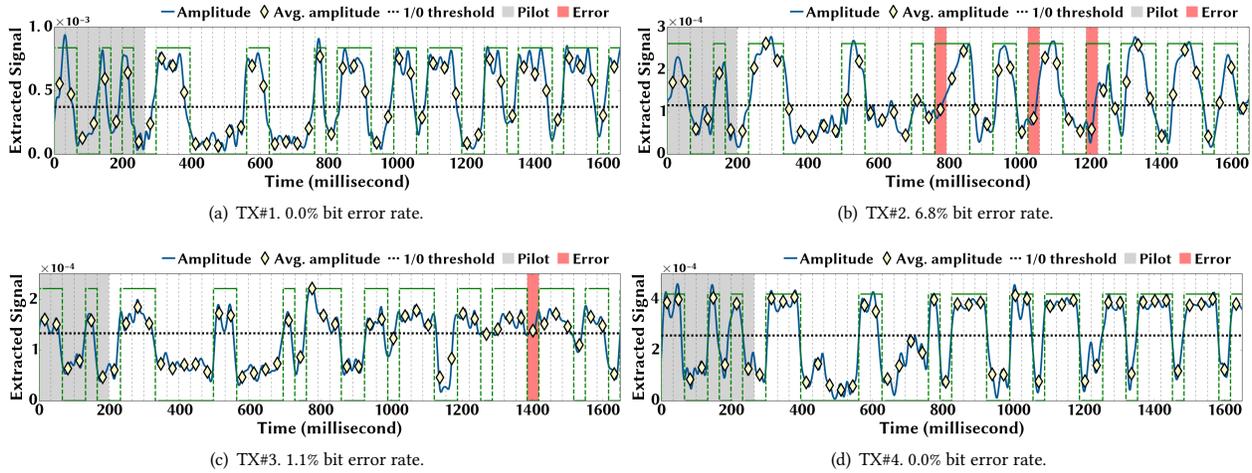


Figure 30: Snapshots of data exfiltration with multiple transmitters.

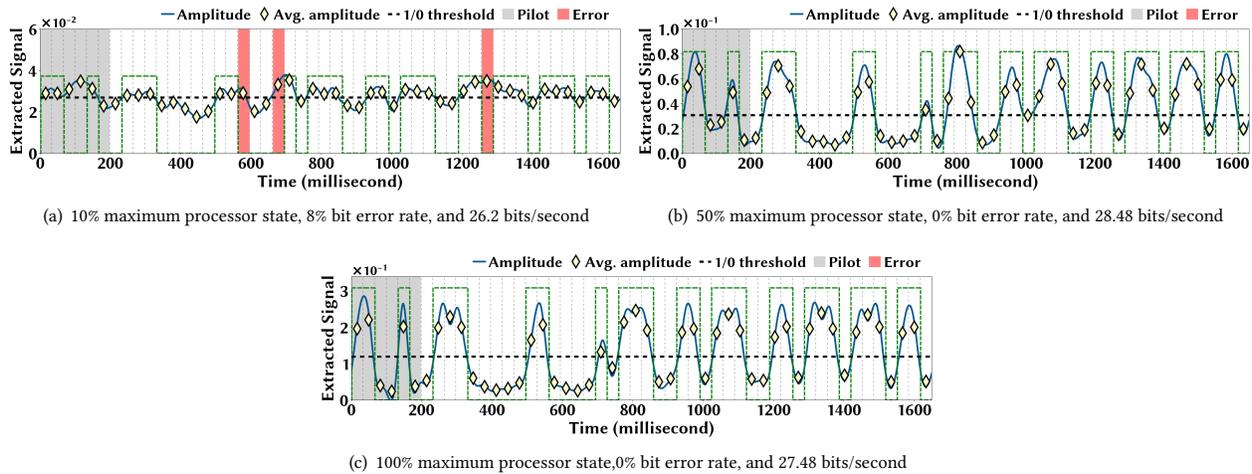


Figure 31: Dell Optiplex computer with different CPU's maximum power states.

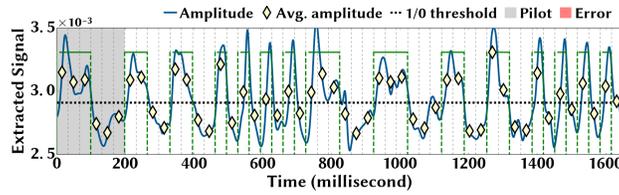


Figure 32: Dell PowerEdge computer in Building B with no line of sight between the receiver and transmitter. The voltage signal is filtered with a passband of $\langle 65.77\text{kHz}, 65.83\text{kHz} \rangle$.

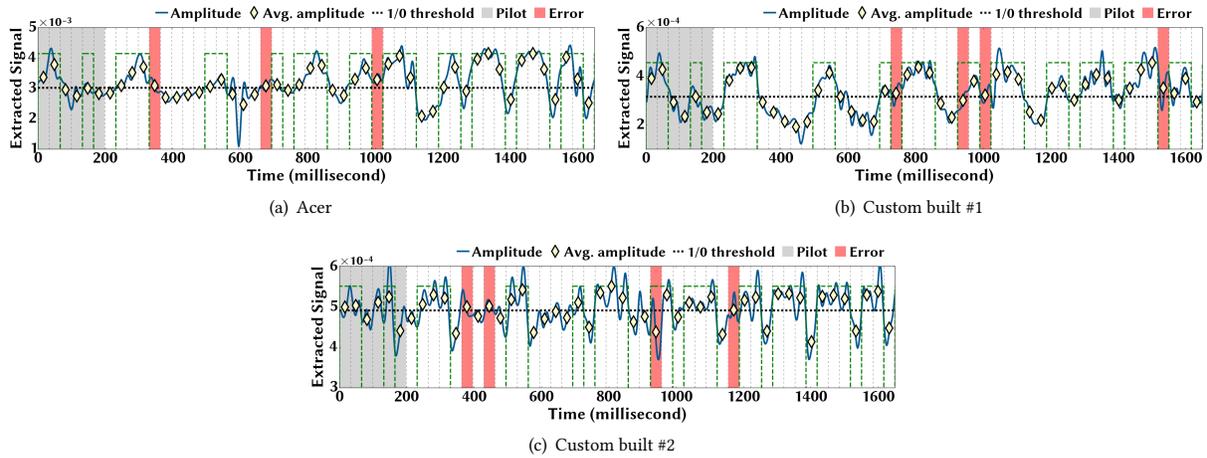


Figure 33: Data exfiltration from different computers.

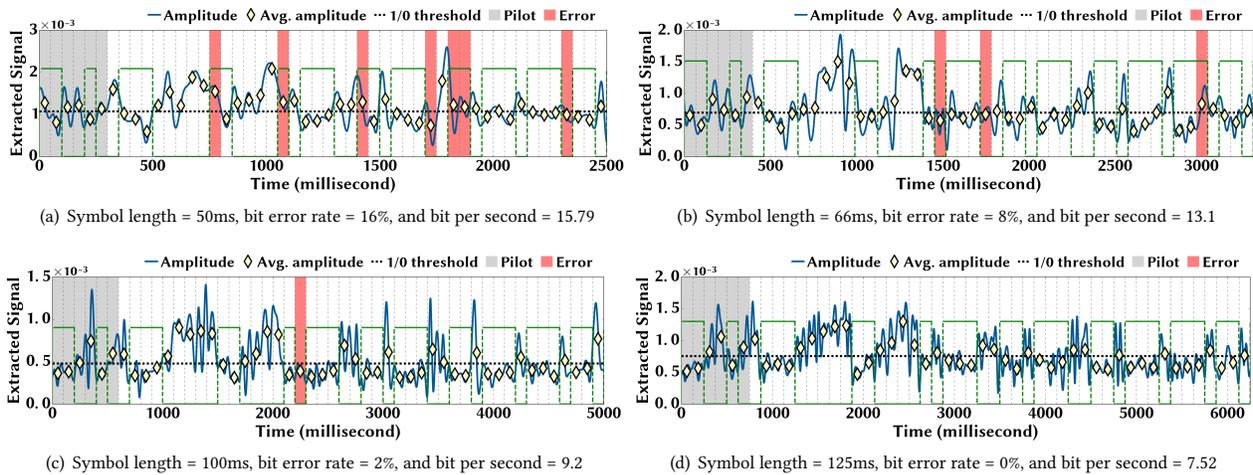


Figure 34: Snapshots of data exfiltration with iMac computer for different bit durations.

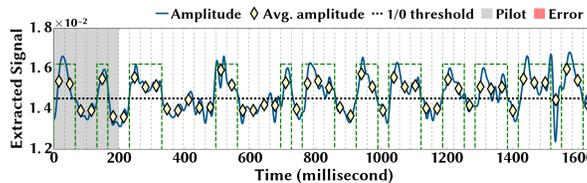


Figure 35: Dell PowerEdge computer powered by a CyberPower UPS. The voltage signal is filtered with a passband of $\langle 65.77\text{kHz}, 65.83\text{kHz} \rangle$. The resulting bit error rate is 0.0% and the effective bit rate is 28.48 bits/second.

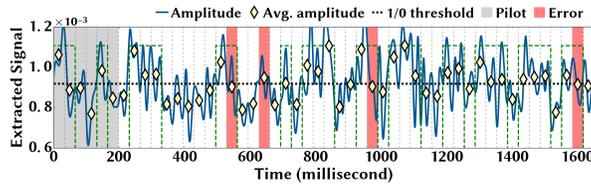
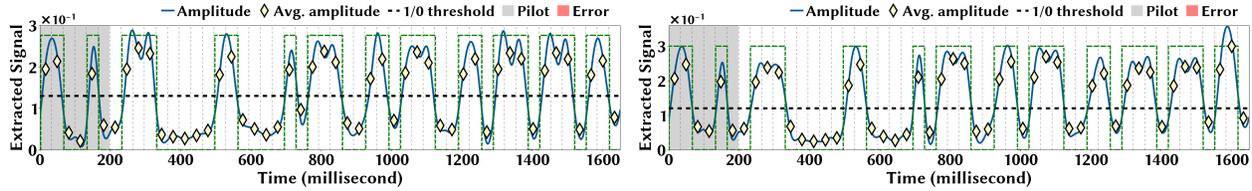
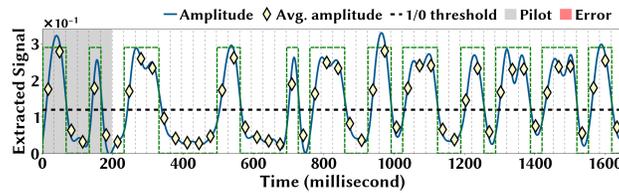


Figure 36: Dell PowerEdge computer with a power line noise filter plugged into the power outlet. The voltage signal is filtered with a passband of <65.78kHz, 65.84kHz>. The resulting bit error rate is 10.2% and the effective bit rate is 25.57 bits/second.



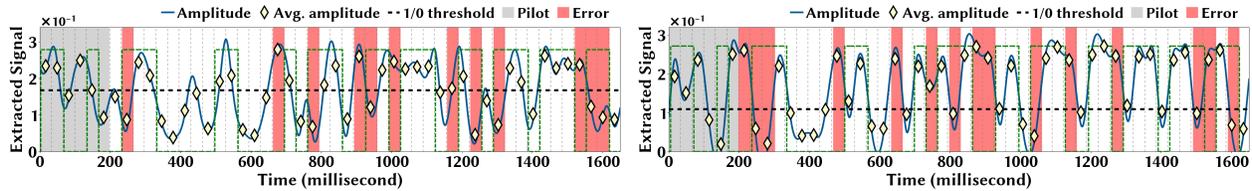
(a) Symbol length = 15 ms, high CPU load = 20% times, bit error rate = 0%

(b) Symbol length = 33 ms, high CPU load = 20% times, bit error rate = 0%



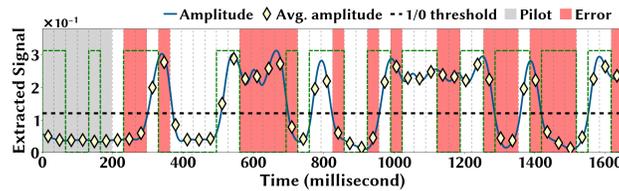
(c) Symbol length = 66 ms, high CPU load = 20% times, bit error rate = 0%

Figure 37: Impact CPU loading intervals with CPU loads 20% of the times..



(a) Symbol length = 15 ms, high CPU load = 60% times, bit error rate = 20%

(b) Symbol length = 33ms, high CPU load = 60% times, bit error rate = 30%



(c) Symbol length = 66 ms, high CPU load = 60% times, bit error rate = 30%

Figure 38: Impact CPU loading intervals with CPU loads 60% of the times..

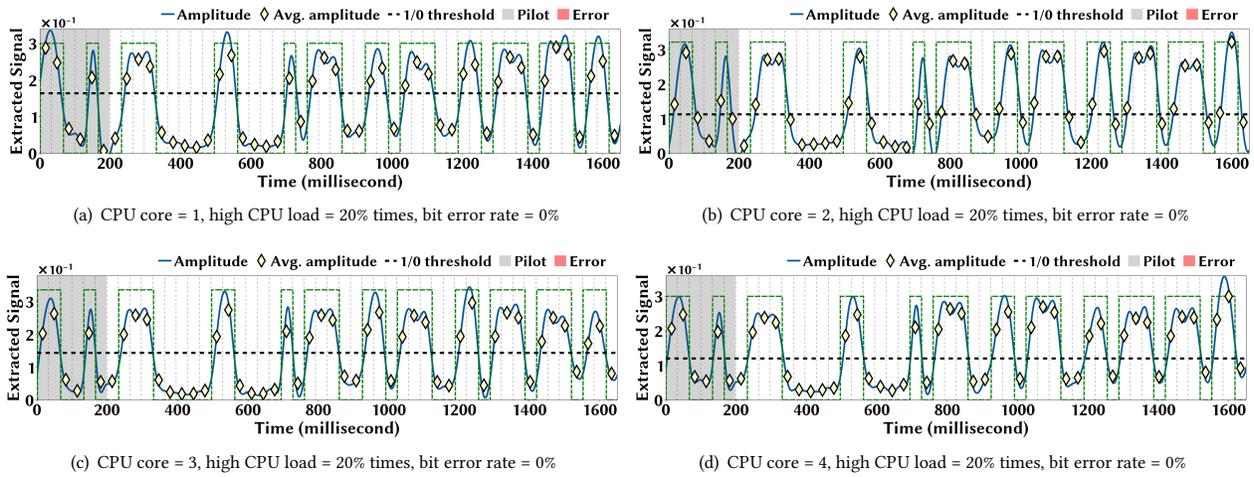


Figure 39: Impact of number of cores used by the defense program with CPU loads 20% of the times.

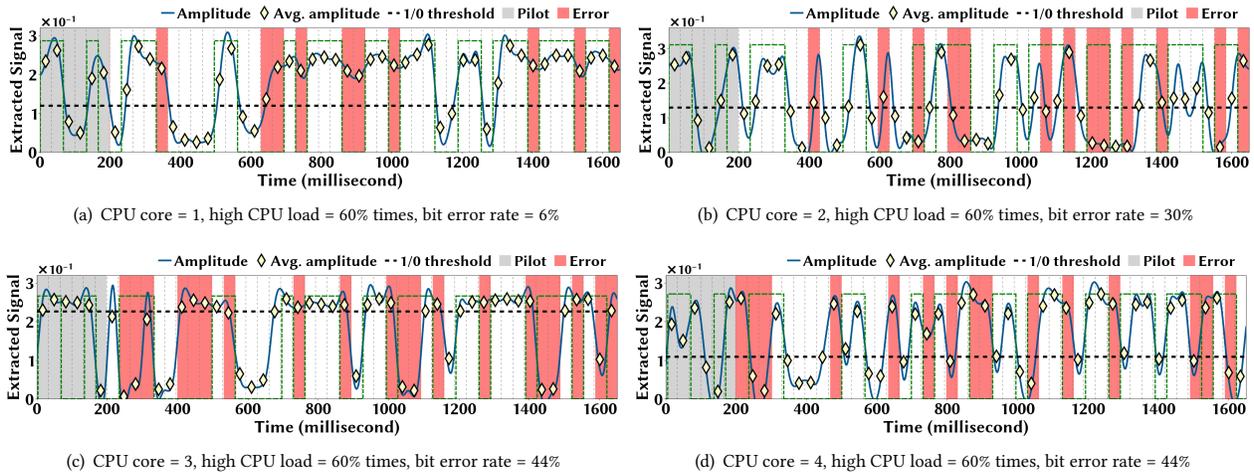


Figure 40: Impact of number of cores used by the defense program with CPU loads 60% of the times